

AD-A147 666

ADAPTING LOGISTICS MODELS TO A MICROCOMPUTER FOR  
INTERFACE WITH COMPUTER... (U) AIR FORCE INST OF TECH  
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST..

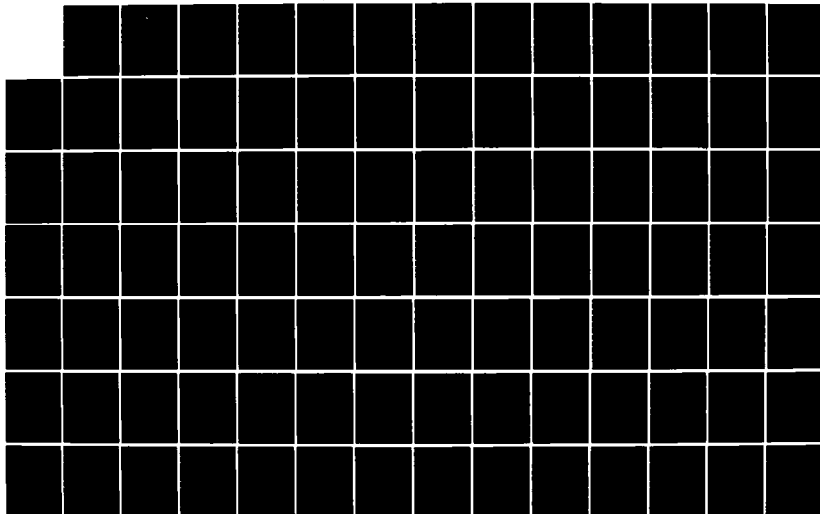
1/3

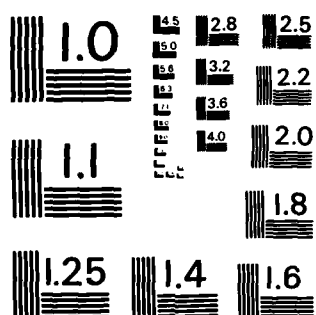
UNCLASSIFIED

D G DAVIDSON ET AL. SEP 84

F/G 15/5

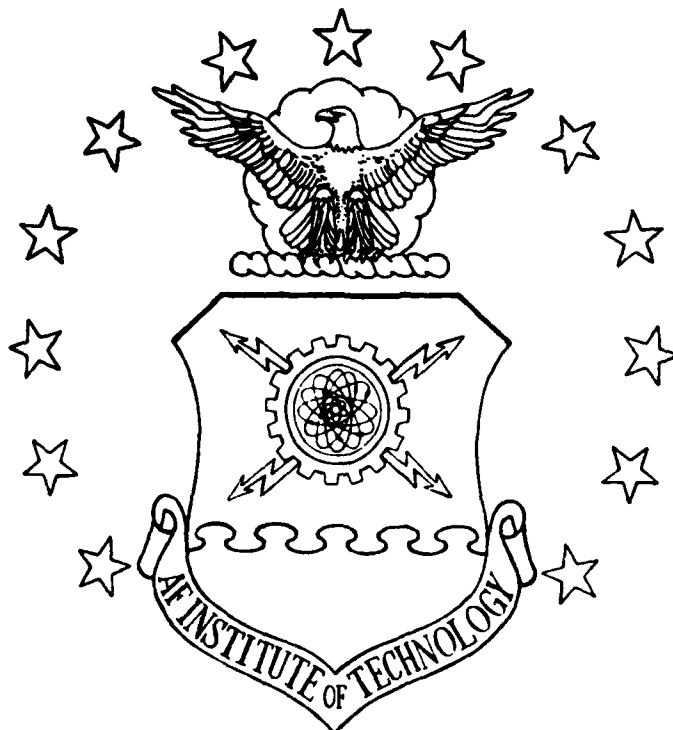
NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A147 666



ADAPTING LOGISTICS MODELS TO A  
MICROCOMPUTER FOR INTERFACE WITH  
COMPUTER-AIDED DESIGN SYSTEMS

THESIS

Donald G. Davidson  
Captain, USAF

John J. Fraser  
Captain, USAF

AFIT/GLM/LSM/84S-11

This document has been approved  
for public release and sale; its  
distribution is unlimited.

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

REPRODUCED AT GOVERNMENT EXPENSE

11 14 000

DTIC FILE COPY

NOV 16 1984  
A

AFIT/GLM/LSM/84

ADAPTING LOGISTICS MODELS TO A  
MICROCOMPUTER FOR INTERFACE WITH  
COMPUTER-AIDED DESIGN SYSTEMS

THESIS

Donald G. Davidson  
Captain, USAF

John J. Fraser  
Captain, USAF

AFIT/GLM/LSM/84S-11

NOV 16 1984

A

Approved for public release; distribution unlimited

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information are contained therein. Furthermore, the views expressed in the document are those of the author(s) and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the United States Air Force, or the Department of Defense.

DATE  
COPY  
PREPARED

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

AFIT/GLM/LSM/84S-11

ADAPTING LOGISTICS MODELS TO A  
MICROCOMPUTER FOR INTERFACE WITH  
COMPUTER-AIDED DESIGN SYSTEMS

THESIS

Presented to the Faculty of the School of Systems and Logistics  
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Logistics Management

Donald G. Davidson, B.G.S

Captain, USAF

John J. Fraser, B.S., M.A.

Captain, USAF

September 1984

Approved for public release; distribution unlimited

### Acknowledgments

The authors wish to express their appreciation to Dr. William B. Askren for his guidance and patience during this research. His assistance and direction greatly aided the authors in defining, narrowing, and developing our research. We would also like to thank Dr. Stephen Denny, Associate Professor of Management Science at Wright State University, for his assistance and guidance. Finally, a special thanks to Dr. Robert B. Weaver for proofreading the drafts and ensuring our logic was logical.

## Table of Contents

	Page
Acknowledgments . . . . .	ii
List of Tables . . . . .	v
Abstract . . . . .	vi
I. Introduction . . . . .	1
Background/Justification. . . . .	1
Problem Statement . . . . .	5
Why a Micro . . . . .	6
Scope and Limitations . . . . .	7
Research Objectives . . . . .	8
Research Questions . . . . .	8
Summary . . . . .	9
II. Methodology . . . . .	10
Overview . . . . .	10
Literature Review . . . . .	10
Interviews . . . . .	11
Implementation of Two Logistics Models . . . . .	12
Summary . . . . .	13
III. Logistics Modeling . . . . .	14
Overview . . . . .	14
Motivation for Using Models . . . . .	15
Classification of Models . . . . .	16
Critical Variables . . . . .	22
Noncritical Variables . . . . .	22
CAD Variables . . . . .	23
Problems and Limitations . . . . .	24
Decision Function Formulation . . . . .	27
Summary . . . . .	31
IV. Microcomputers . . . . .	32
Overview . . . . .	32
What is the Question . . . . .	32
Hardware Considerations . . . . .	34
Processor Speed . . . . .	36
Memory Addressability . . . . .	39
Numeric Capability . . . . .	40
Input/Output . . . . .	42

Software Considerations . . . . .	44
Operating Systems . . . . .	45
Language Processors . . . . .	45
Summary . . . . .	46
V. Implementation of Two Logistics Models . . . . .	47
Overview . . . . .	47
Model Selection . . . . .	47
Obtaining the Source Code . . . . .	48
What Do They Do? . . . . .	49
Source Code Conversion . . . . .	50
Editing Capabilities . . . . .	50
Nonstandard Statements . . . . .	50
Character Data . . . . .	51
Program Debugging . . . . .	51
Running the Model . . . . .	55
Summary . . . . .	55
VI. Conclusions and Recommendations . . . . .	56
Conclusions . . . . .	56
Logistics Models . . . . .	56
Microcomputers . . . . .	57
Recommendations . . . . .	60
Appendix A: Logistician Interview . . . . .	62
Appendix B: Computer Specialist Interview . . . . .	65
Appendix C: Description of Models . . . . .	68
Appendix D: Disk File Setup . . . . .	72
Appendix E: A Session with RMCM . . . . .	74
Appendix F: Program Listing . . . . .	84
Bibliography . . . . .	188
Vita . . . . .	192

### List of Tables

Table	Page
I. Logistics Model Variables . . . . .	26
II. IBM-PC versus Minicomputer Comparison . . . .	37
III. Microprocessors Memory Capacities in Bytes . .	40
IV. Floating-Point Execution Speed in Microseconds . . . . .	42
V. Comparision of Single and Double Precision Calculations . . . . .	54
VI. Glossary File Search Times . . . . .	58

Abstract

Logistics concerns such as reliability and maintainability are the results of product design. Logistics models are the tools used by the logistics engineers to analyze these logistics concerns. Currently, logistics models are run primarily on mainframe computers and at later stages of the design process. If logistics models were adapted to microcomputers, the models would be more accessible to the logistics engineers, thus resulting in products which are more reliable and more easily maintained.

A further step would be to interface these models with a computer-aided design (CAD) system. CAD systems have proven to be a very useful engineering tool during product design. The interfacing of these models to a CAD system would allow the logistics engineer to analyze design earlier, thus achieving greater flexibility in the design process.

This research examines the difficulties of selecting models for incorporation into a CAD system and the use of microcomputers to run these models. A selection function was developed to identify models for specific types of analysis and their suitability for incorporation into a CAD system. The literature on microcomputers was examined to determine the limitations of microcomputers to run large logistics

models. To further define these limitations the Reliability Maintainability Cost Model was adapted to an IBM-PC micro-computer.

# ADAPTING LOGISTICS MODELS TO A MICROCOMPUTER FOR INTERFACE WITH COMPUTER-AIDED DESIGN SYSTEMS

## I. Introduction

### Background/Justification

During the last ten years the engineering design process has been changing. Several key developments have played a major role during this transitional period. Engineering News-Record (December 1981) identifies the development of Computer-Aided Design systems as one of the most significant events in the history of the design process (11). The design engineer translates the product requirements into a hardware design. The product design sets the upper limit of performance. Performance cannot be manufactured; therefore, design engineers spend countless hours trying to insure a product is properly designed before it goes into production.

Computer-aided design (CAD) is a tool, developed over the last ten years, which has aided engineers in their quest for increasing product performance. An engineer using CAD can now make numerous design changes to a proposed product and run several engineering analysis programs on the new design within minutes. This increases the total number of designs the engineer can study thus increasing the likelihood

of a favorable design being developed. Other benefits besides improved product design are also realized when a CAD system is used in the design process (24:233):

- The designer is aided in solving design problems that are not easily tackled without a computer,
- Reduced design costs are realized through a reduction in manhours,
- Companies are able to reach their full productive potential by avoiding bottlenecks in the design office, and
- Product design time is reduced.

The next major event to impact the design process was the realization that logistics concerns such as reliability, supportability, and spares requirements are also the result of product design; and like performance, they cannot be manufactured but must be designed into a product. With this realization came the development of Logistics Support Analysis (LSA). "LSA efforts, to be of maximum effectiveness, should commence when system/equipment concepts are being formulated [34:19]." Blanchard (4:14) offers the following description of LSA:

LSA constitutes the application of selected quantitative methods to (1) aid in the initial determination and establishment of logistics criteria as an input to system design, (2) aid in the evaluation of various design alternatives, (3) aid in the identification and provisioning of logistics support elements, and (4) aid in the final assessment of the system support capability during consumer use. LSA is a design analysis tool employed throughout the early phases of system development and often includes maintenance analysis, life-cycle costs analysis, and logistics modeling.

"Up-front analysis and design is imperative if the resulting design is to be supportable in a cost-effective manner [23:1]." This requires the logistics engineer to be a part of the product design from the earliest possible moment. The LSA system is a step in the right direction and a tremendous improvement over the previous system; but LSA also has several major shortcomings (39:2-5):

- Creation of redundant data bases increases development costs and chances for data error and
- Analysis of the data and communication of the findings to the design engineer often occur too late to allow for effective design changes.

CAD technology has the potential to overcome the drawbacks of LSA and "could very well become the next-generation tool for both maintainability and logistic support analysis [23:1]. During design of an item on a CAD system, data regarding the design is captured and stored in a data base. This data is then available for all interested parties to use and the need for duplicate data bases is eliminated.

The current iterative LSA process would also be brought a step closer to the design engineer. A logistics engineer, for example, could analyze, on a timely basis, a new product design for maintainability if the appropriate computer models were available and interfaced with the CAD system. This ability would aid the logistics engineer in achieving his goal:

. . . to evaluate the overall system, define the elements involving high risks, conduct trade-off studies to evaluate the risks (or various design approaches), and document all data so that all elements of design and support can be tailored to produce an optimum overall system [4:18].

Additional benefits would be achieved also if logistics models were interfaced with CAD. "The logistics engineer is considered part of the design team; but normally, he becomes no more than a data collector and his role as engineer is ignored (40:2)." A logistics interfaced CAD system would help the logistics engineer in the performance of his data collection task.

Research by the Air Force Human Resources Laboratory at Wright-Patterson Air Force Base offers the following reason to interface logistics models with a CAD system:

If maintenance and logistics requirements can be incorporated into the automated (CAD) engineering design process, design for maintenance will become technically, organizationally, and motivationally a standard practice [37:2].

Lockheed Missile and Space Company already has begun to use CAD to determine maintainability of design. The maintainability engineer operates ADAM (Anthropometric Design Aid Manikin) from a CAD terminal.

ADAM identifies maintenance interfaces and establishes relative scale and technician working positions which help the designer to more fully appreciate the maintainability implications of his designs. The use of ADAM does not interfere with the engineer's prerogatives, but provides a realistic basis for discussing the best means of meeting maintenance requirements such as access, reach, and working postures [8:12].

### Problem Statement

The question now becomes how does one go about interfacing logistics models with CAD. Several alternatives exist. One could specifically design logistics models and tools (such as Lockheed's ADAM) to run on a CAD system. This poses several drawbacks:

- The cost of providing or insuring adequate access to a CAD terminal for each logistics engineer could prove prohibitive.

- "The inherent lack of portability of CAD programs from one host machine to another is a major factor in the limited availability to users [26:4]." This would limit the logistics engineer to running only those models implemented on the CAD system which holds the design data.

- The CAD system's computing power could quickly become overburdened. "With many users vying for limited resources, a user's turnaround time to complete a particular design problem can be significantly degraded [26:4]."

The interfacing of a CAD system with an alternate computing source offers a more flexible response:

- The CAD system's computing power would be taxed only when transferring required data to and from the alternate computing source.

- If a communication link was used in the interface, the alternate computing source could tap into several CAD systems, thus allowing the logistics engineer to analyze each

design using any logistics model available.

The Air Force Human Resources Laboratory at Wright-Patterson Air Force Base is currently studying the problems of interfacing an alternate computing source with a CAD system. The initial thrust of this study has focused in two areas:

1. How does one interface with a CAD system and its data base?
2. What problems and constraints are encountered if a microcomputer is used as the alternate computing source? It is the intent of this thesis to investigate and answer the second question.

#### Why a Micro?

The selection to use a microcomputer was based on several factors:

- People have experienced problems in using large timesharing systems on mainframe computers (3:112; 26:4).
- The increasing computational power coupled with the decreasing cost of microcomputers offers an attractive alternative to the use of a large mainframe or minicomputers. The computational power of microcomputers is discussed in Chapter 4.
- Microcomputers have introduced a new dimension to CAD. No longer are CAD systems run only on large mainframe computers or dedicated processors. For example, CAD systems now exists for the Apple computer (CAD-Apple System by T&W

Systems) and the IBM-PC (MicroCAD by Computer Aided Design, AutoCAD by Autodesk, Inc., and The Drawing Processor by BG Graphics). Considering that Intel and National Semiconductor have already begun to market 32-bit microprocessors (Intel: iAPX432 and National Semiconductor: 16032) that approach mainframe computational power, it should only be a short period before microcomputers support multiuser CAD systems.

- With the migration of CAD systems to microcomputers, a distributed processing environment would allow a natural interface to exist between the design engineer's workstation and the logistics engineer's workstation.

- Cost of a computer system is an important factor. An LSA computer system described by Naas and Eames (29) costs approximately \$25,000. Twenty-five thousand dollars would purchase five to six microcomputer configurations described in the following section.

#### Scope and Limitations

For the purpose of this research a microcomputer was defined to consist of the following hardware:

1. Intel's iAPX 88/10 (8088) microprocessor,
2. Intel's iAPX 88/20 (8087) numeric data processor,
3. 512K random access memory,
4. graphics capability,
5. ten megabyte fixed disk,
6. one floppy disk, and
7. communications gear for operation at a minimum

speed of 1200 baud.

By defining minimum hardware requirements, we insure upward compatibility with the majority of computing systems available.

### Research Objectives

An underlying theme of this thesis is the improvement of the logistics analysis process during the design of a product. We do not want to identify problems with implementing just any model on a microcomputer but only those logistics models which would be useful during the design stage of a product; therefore, the overall objective of this research is twofold:

1. Identify models considered useful by logistics engineers during the design stage of a product.
2. Identify constraints and problems one encounters in the implementation of these models on a microcomputer.

### Research Questions

To achieve the first research objective the following questions must be answered:

1. In the early design stages, what type of analysis does the logistics engineer want to perform?
2. What problems are associated with the use of logistics models in the early design stages?

To achieve the second research objective the following questions must be answered:

1. What are the hardware constraints imposed by

the use of a microcomputer?

2. What are the software constraints imposed by the use of a microcomputer?

### Summary

This chapter has presented a brief introduction to the subject of this research effort. Justification for the study has been given and objectives identified. The remaining chapters present the methodology used in achieving these objectives, documentation of the results, and conclusions and recommendations.

## II. Methodology

### Overview

The validity of any research is often questioned and this is only reasonable. Research delving into a new area is subjected to even harsher scrutiny. It is with these thoughts in mind that the methodology of this research effort was developed. The methodology must not only meet the objectives of this research but also support the validity of conclusions put forward. To achieve this goal a methodology of three parts was developed:

1. Literature review
2. Interviews
3. Implementation of two logistics models.

### Literature Review

A literature review was conducted in two parts. The first part of the literature review attempted to determine which logistics models should be interfaced to a CAD system. It was felt this was an important first step because one could be quickly overwhelmed by the number of models available to study. We also assumed that the problems encountered in implementing just any logistics model on a microcomputer were probably not the same as implementing models dealing with product design.

The second part of the literature review focused on microcomputers. This review of microcomputers was directed

in three areas. First, the current capabilities of microcomputers were discussed. Answers to the following questions were sought:

- Does the processing speed of microcomputers make it impractical to implement the desired models?
- Do microcomputers possess the required numeric capability required by the desired models?
- Do microcomputers have enough memory expandability to allow implementation of the desired models?
- Do microcomputers support the peripheral devices required by the desired models?

Next, the current state of software development was considered. Answers to the following questions were sought:

- What problems have others encountered in implementing programs on microcomputers?
- Are the operating systems currently available for microcomputers sophisticated enough to support concepts such as program overlays and memory paging?
- Are the programming languages currently available for microcomputers sufficiently developed to make programming practical?

### Interviews

Structured interviews were conducted with individuals knowledgeable in microcomputer software development and with logistics engineers. The purpose of the interviews was to confirm and enhance information found in the literature

regarding the questions listed above.

Separate interview instruments were developed for each topic of discussion (refer to Appendices A and B). Each instrument was validated by conducting two trial interviews with individuals knowledgeable with the subject material.

#### Implementation of Two Logistics Models

To further define and refine the problems and constraints that are encountered when using a microcomputer to implement logistics models, two logistics models were selected for implementation. The models selected were identified by logistics engineers as useful for performing analysis during a product's design stage. During the implementation process, specific attention was paid to the following factors:

- What input data would be provided from the CAD data base versus what information would be provided by the logistics engineer?

- The technical feasibility of implementing the models (for example, do the algorithms used by the models require modification to allow the models to be implemented on the microcomputer?)

- What type of programming effort was required to implement the models?

- Do the models provide results comparable to the results from previous host computers?

- Do the output of the models lend itself to a microcomputer application?

### Summary

This chapter presented the methodology used in developing the criteria for model identification. The topic of this research called for a methodology that was not based upon quantitative rigor but a data gathering synthesizing approach. The relative lack of research data in this area also dictated a methodology based upon what many would call common sense. The next two chapters present the information gathered from the literature and interviews.

### III. Logistics Modeling

#### Overview

We have have put forth the concept that involving the logistics engineer earlier in the design process will enhance the overall design of a system. This is not a novel concept but one that is readily supported in the literature.

Consider the following statements:

Benjamin Blanchard:

. . . experience has indicated that a great deal of impact on the projected life-cycle cost for a given system or product stems from decisions made during the early phases of advance system planning and conceptual design [4:5].

Terrance Sterkel:

Logistics driven design offers cost effectiveness, minimum technical risks, and adherence to system requirements [40:1].

Edward Naas and Susan Eames:

Logistics must be considered as a design parameter. We must influence the design with relative quickness so that the end item can be maintained and supported [29:1].

What is interesting is that each of these individuals offers a different viewpoint as to why the logistics engineer should be involved in the design process, but they all support two common themes:

1. The logistics engineer should become involved in the design process as early as possible.
2. Logistics models provide the logistics engineer the tools to analyze a system or product's design from several different perspectives.

The second of these two themes is explored in this

chapter. We begin by offering some motivation for the use of logistics models. We then focus on the classifications of models and the inherent problems/limitations of using these models. Lastly, we developed an evaluation technique for identifying logistics models suitable for incorporation into a CAD system.

#### Motivation for Using Models

The benefits derived from using logistics models are numerous. Perhaps the most important benefit derived from the use of models is that the logistics engineer is allowed to analyze the design of a system from several different perspectives "before substantial money and time is spent on its development [32:vi]."

Models also allow the logistics engineer to deal more easily with the increasing complexities of designing, procuring, and maintaining a system (14:1).

A more subtle benefit derived from the use of models is the aid in repetitive analysis. Once a model is designed and implemented, it aids the logistics engineer by reducing the time required to analyze several design trade-offs (35:27).

Blanchard offers the following six reasons why models are useful (4:403):

1. Models allow numerous interrelated elements of a system to be treated as a whole.
2. Models allow many different alternatives and variations of the system to be studied.

3. Models aid in solutions to problems that otherwise might not be solvable.
4. Models aid in identification of high-risk areas.
5. Models aid in processing large amounts of data efficiently.
6. Models aid in assessing the impact of alternative actions on the total system.

To be fair, it should be noted that the use of logistics models is not all rosy in that many models are not properly verified or cannot accept estimated data.

#### Classification of Models

With so many benefits possible, it is no wonder that the use of models throughout the logistics community has been increasing. Several of the applications in which models are used to perform analysis are (14:2):

- life cycle cost,
- maintenance repair policy,
- spare parts purchase and distribution,
- resource usage and requirements,
- maintainability design,
- reliability design, and
- supportability design.

By no means is this an exhaustive list. If one couples the numerous applications in which models can be used with the proliferation of models, one becomes faced with the complex task of selecting the proper model for the job. This problem

is even further compounded by the lack of a consistently used scheme to classify models. While it was not the intent of this thesis to develop a model classification scheme, it was necessary for us to delve into this question to be able to identify models that are appropriate for use in the early design stages of a system.

Perhaps the easiest way to classify models is by their solution technique: simulation, mathematical, statistical, networks, and miscellaneous (1:5-6). One can further break down these main categories:

- simulation
  - continuous change models: make use of fixed time increments as the timing mechanism in the model.
  - discrete change models: make use of the next event concept as the timing mechanism in the model.
- mathematical programming
  - linear programming
  - integer programming
  - nonlinear programming
  - goal programming
  - dynamic programming
- statistical
  - regression analysis
  - exponential smoothing
  - sampling
  - hypothesis testing

- networks

- PERT

- CPM

- miscellaneous

- queuing theory

- inventory theory

- costing/accounting theory

- heuristics

"A particular methodology will probably be dominant rather than exclusive in a model [32:12]" but the solution technique needs to be known by the model user. Different solution techniques could and probably would yield different results when applied to the same problem, thus the motivation for classifying models by solution technique.

Classifying models by solution technique has several drawbacks as does any classification scheme. One particularly large problem created by this classification scheme is that it offers no insight into when the model should be used for analysis or the function of the model. Thus we have classification schemes for models by life cycle phase and by application.

Blanchard has broken the life of a system into six distinct phases (4:317):

- Conceptual phase: the need for the system is defined.

- Advance development phase: system configuration

is defined.

- Detail design and development phase: the systems and all their support items are designed and tested.
- Production and/or construction phase: the system and its support items are produced.
- Operational use phase: the system is deployed.
- System retirement phase: the system is phased out.

Models used in one phase may or may not be suitable for use in another phase due to data requirements (availability and validity) or analysis performed; therefore, it is imperative for the logistics engineer to know what models apply at a particular stage of a system's life.

There are several classification schemes for categorizing models by application. Paulson, Waina, and Zacks (32) used this scheme in classifying 46 models into seven applications categories: spares, aerospace ground equipment (AGE), personnel, maintenance posture, operations, life cycle cost, and project management. The Defense Logistics Studies Information Exchange (DLSIE) produces catalogues of logistics models using five major categories but provides a further breakdown by a secondary coverage index to provide a total of 49 categories:

<u>Major Category</u>	<u>Number of Sub-Categories</u>
Accounts	3
Facilities	7
Material	22
Military Operations	12
Personnel	5

The problem that arises from using just any one of the

classification schemes by itself is that the logistics engineer has no immediate means to identify a model that is used in the specific phase of a system's life cycle and which performs a particular type of analysis and which uses a particular solution technique.

Interviews conducted suggest that logistics engineers use only several factors in determining which model to use for analysis. Not surprisingly, which factors are considered important varies among logistics engineers (for example, ease of use, availability, and solution technique).

We choose to represent the appropriateness of a model for performing the desired analysis as a function of  $n$  variables:

$$R_i = f(x_{i1}, x_{i2}, \dots, x_{in}) \quad \text{for the } i\text{'th model} \quad (1) \\ \text{being considered}$$

Value  $R_i$  represents a rating assigned to the  $i$ 'th model and each variable  $x_{ij}$  represents various characteristics of the model that the logistics engineer considers important. There exist several methodologies for implementing a function of this form.

First, one could describe the set of characteristics for each model under consideration and apply the function to the set. The models yielding a value greater than or equal to a minimum standard rating would then be considered appropriate for performing the desired analysis.

A second method would be to describe the function in

terms of a decision tree where each variable represents a node of the tree. This method becomes very difficult to implement as the number of branches increase.

We can enhance the usefulness of the rating function by redefining the function using two classes of variables: critical and noncritical. The function now takes the form:

$$R_i = f(x_{i1}, \dots, x_{in}, y_{i1}, \dots, y_{im}) \quad (2)$$

Each  $x_{ij}$  represents a critical variable. A critical variable is a characteristic that if not present in the model forces the rating  $R_i$  to zero. Each  $y_{ij}$  represents a noncritical variable. A noncritical variable is a characteristic that makes the use of the model desirable to the logistics engineer, but does not negate the use of the model if not present.

This definition creates the problem of determining which variables are critical and which ones are noncritical. This is not a new problem; as mentioned we found little agreement on this matter in the interviews we conducted. One is also faced with the quantification of these variables. So the function we are describing is not a numerical function but the decision process the logistics engineer uses to select a model to perform the desired analysis. Paulson, Waina, and Zacks accurately describe the situation: "It is incumbent upon the decision maker to be sure he has selected the proper tools for the tasks [32:14]." With this thought in mind we offer the following questions based upon various opinions put

class of variables that pertain to the suitability of using the model in a CAD environment. The new function would have the following form:

$$(R_{i1}, R_{i2}) = f(x_{i1}, \dots x_{in}, y_{i1}, \dots y_{im}, z_{i1}, \dots z_{ik}) \quad (3)$$

The value  $R_{i1}$  represents our original value  $R$  which was defined to be the rating of appropriateness of the model to perform the desired analysis. Value  $R_{i2}$  represents a rating of the suitability of the model to operate in a CAD environment. Each  $z_{ij}$  represents a variable to be used in determining this suitability value. The following questions represent possible variables of the rating function:

CAD Variables.

1. Is the model interactive?
  - If not, can it be made interactive?
2. Are the results of the model representable in graphical form?
3. Is the run time of the model of a short to medium duration (1 to 5 minutes)?
4. Does the model require data from the CAD data base?
5. Is the model compatible with the CAD hardware and software?

Three major ideas should be extracted from the preceding discussion:

- The number of variables that come into play in the selection of a model to perform a particular type of analysis

is large.

- The varying classification schemes focus only upon one major aspect of a model at a time, thus hindering the logistics engineer in his decision process of selecting a model.

- There exist wide and varying opinions about which characteristics of a model should be used in the selection process.

With these ideas imparted we now turn our attention to some of the problems and limitations of using models.

#### Problems and Limitations

A model, whether a logistics or engineering model, is an abstraction of reality; therefore, inherent problems and limitations are associated with its use. One of the most limiting factors encountered is the data the model processes.

Data requirements vary greatly from model to model . . . . . Whatever its requirements, however, a model's effectiveness is dependent upon the quantity and quality of the data available to drive it [32:7].

Generally, the further the system has progressed through the design process the more data is available to process. The accuracy of data also improves. As we attempt to move in the other direction (earlier in the design stages), data availability and accuracy become limiting factors the model must be able to handle. Fortunately, methodologies are being developed to aid the logistics engineer in overcoming these problems. For example, a logistics risk assessment methodology

described by Wood (44) establishes data estimates and assesses associated risks of using these estimates. Wood offers the following motivation for using such methodologies (44:1):

The objectives of a logistics risk assessment methodology are to: (1) generate an acceptable and supportable point estimate rather than a composite of subjective estimates which is difficult to justify, (2) provide management with an associated risk assessment of given point estimate, (3) provide a consistent methodology for adjusting the initial point estimates as hardware definition reduces the ranges of realistic parameters, and (4) provide a computerized model for evaluating complex interrelationships contributing to resultant point estimate.

It should be noted that the availability of only estimated data should not stop one from using models for analysis. The user needs only to understand the limitations imposed by the data.

Even when the uncertainty cannot be easily quantified, models can be used to explore system design/support cost interactions and thus define at least the desirable ranges of system parameters [32:47].

Another problem associated with using models is the requirement for the logistics engineer to understand how the model treats a particular variable. For example, if an output is computed from data estimates, the output will be an estimate of less precision (5:27). Considering the number of variables possible this can be a large task. Table I gives a good indication of the numerous variables a model may process.

TABLE I

Logistics Model Variables (32:8)

Item	Repair points
Unit Cost	Manhours to repair
Reliability	Maintenance skills
Weight	Parts cost/repair
Volume	Labor rate
Procurement lead time	Repair cycle length
Procurement cost	Order and shipping time
R and D cost	NRTS rate
	Condemnation rate
System	Distance for next echelon
Program for utilization	Packing cost
Geographic deployment	Shipping cost
Force size	AGE cost (acquisition, installation, O and M)
Force life	AGE weight and volume
OR rate	AGE quantity
On-equipment maintenance cost	Facilities cost
Training cost	Technical data pages
Interest rate	Technical data cost
Stock points	
Spares level	Maintenance postures
Supply effectiveness	
Supply administration cost	
Reorder policy	

A problem also arises in data compatibility. Data may be available but not in a form usable by the model. Processing the data to obtain the correct form could be expensive or even worse cause the accuracy of the data to be diluted.

Besides data problems, Blanchard offers two additional problems stemming from the use of models (4:405):

1. There is a chance the logistics engineer may become so enamored with the model that his concept of the problem becomes distorted.

2. Models are only tools and not a substitute for experience and judgment.

### Decision Function Formulation

With the conceptualization of the decision process behind us, it is now time to turn our attention to a generalized formulation of this function. The decision function takes the following form:

$$(R_{i1}, R_{i2}) = f(x_{i1}, \dots, x_{in}, y_{i1}, \dots, y_{im}, z_{i1}, \dots, z_{ik})$$

$$= \begin{cases} R_{i1} = \begin{cases} 0 & \text{if } x_{ij} \cap X_j = \emptyset \\ & \text{for any } j = 1, 2, \dots, n \\ \text{else} \\ \sum_{j=1}^M a_j c_j & \text{where } c_j = \begin{cases} 1 & \text{if } y_{ij} = \text{Yes} \\ 0 & \text{else} \end{cases} \end{cases} \\ R_{i2} = \sum_{j=1}^K b_j c_j & \text{where } c_j = \begin{cases} 1 & \text{if } z_{ij} = \text{Yes} \\ 0 & \text{else} \end{cases} \end{cases}$$

where

$R_{i1}$  = the appropriateness rating of the  $i$ 'th model

$R_{i2}$  = CAD adaptability rating of the  $i$ 'th model

$x_{ij}$  = set of attributes representing the  $j$ 'th critical variable of the  $i$ 'th model

$y_{ij}$  = attribute representing the  $j$ 'th noncritical variable of the  $i$ 'th model

$z_{ij}$  = attribute representing the  $j$ 'th CAD variable of the  $i$ 'th model

$X_j$  = set of allowable values for the  $j$ 'th critical variable

$a_j$  = weight given to the  $j$ 'th noncritical variable

$b_j$  = weight given to the  $j$ 'th CAD variable

A specific example will aid in understanding how this function can be used. The following example is aimed at identifying a model that is suitable to be used in the conceptual design phase and at the same time lends itself for incorporation into a CAD. Data used in this formulation is taken from Appendix C. Appendix C gives a summary the two models considered. The data listed for each model was synthesized from reports about the particular model and is not intended to serve as a detailed description of the capabilities of the model. It also should be noted that one is not limited to using only the variables listed below but can define any number of variables in any class for which data can be gathered for each model.

The critical variables and allowable values for these variables were defined as follows:

1.  $x_{i1}$  represents the set of life cycle phases to which the  $i$ 'th logistics model is appropriate and
2.  $x_{i2}$  represents the set of the types of analysis the  $i$ 'th logistics model performs.
3.  $x_1 = \{\text{Conceptual Phase}\}$
4.  $x_2 = \{\text{Reliability Analysis, Spares Analysis, Life Cycle Cost Analysis, and Maintainability Analysis}\}$

Noncritical variables were defined as follows:

1.  $y_{i1}$  represents the answer to the question: Is the solution technique appropriate to the type of analysis being performed?

2.  $y_{i2}$  represents the answer to the question: Is data available in this life cycle phase to run the model?

3.  $y_{i3}$  represents the answer to the question: Will the model accept estimated data?

4.  $y_{i4}$  represents the answer to the question: Is estimated data suitable to the level of analysis being performed?

5.  $y_{i5}$  represents the answer to the question: Is the model's output in a form usable to the logistics engineer?

6.  $y_{i6}$  represents the answer to the question: Does the model perform multiple types of analysis?

7.  $y_{i7}$  represents the answer to the question: Can the model operate without special peripheral units?

8.  $y_{i8}$  represents the answer to the question: Does documentation for the model exist?

CAD variables were defined as follows:

1.  $z_{i1}$  represents the answer to the question: Is the model interactive?

2.  $z_{i2}$  represents the answer to the question: Can the model be made interactive?

3.  $z_{i3}$  represents the answer to the question: Are the results of the model representable in graphical form?

4.  $z_{i4}$  represents the answer to the question: Is the run time of the model of a short duration?

5.  $z_{i5}$  represents the answer to the question:  
Will the model accept input from the CAD data base?

6.  $z_{i6}$  represents the answer to the question:  
Will the model run on the CAD system or an interfaced micro-computer?

To simplify calculations for this example each  $a_j$  and  $b_j$  was defined as one. Using the data from Appendix C for the Reliability Maintainability Cost Model (RMCM) and the Network Repair Level Analysis Model (NRLA), we can obtain the values for each variable.

Variable	RMCM	NRLA
$x_{i1}$	Conceptual	Development
$x_{i2}$	Reliability	Level of Repair
$y_{i1}$	Yes	Yes
$y_{i2}$	Yes	No
$y_{i3}$	Yes	Yes
$y_{i4}$	Yes	No
$y_{i5}$	Yes	Yes
$y_{i6}$	Yes	No
$y_{i7}$	Yes	Yes
$y_{i8}$	Yes	Yes
$z_{i1}$	Yes	No
$z_{i2}$	Yes	Yes
$z_{i3}$	Yes	Yes
$z_{i4}$	No	No
$z_{i5}$	Yes	Yes
$z_{i6}$	Yes	Yes

Applying the decision function (page 28) to the above values, the function yields a value of (8, 5) for RMCM compared to a value (0, 4) obtained for NRLA. The first value indicates that RMCM satisfies the critical variables and satisfies many of the noncritical variables. The first value for NRLA indicates the model does not meet all the critical variables.

The next value for each of the two models indicates each model has potential for interfacing with a CAD system.

#### Summary

We presented in this chapter what we believe is an accurate representation of some of the difficulties a logistics engineer faces in selecting and using logistics models. We also presented a method for empirical evaluation of the appropriateness of a model to perform the desired analysis and to identify models suitable for incorporation into a CAD system.

We now turn our attention to the topic of microcomputers to better understand their capabilities and limitations.

#### IV. Microcomputers

##### Overview

The microcomputer offers the logistics engineer a chance to increase productivity just as the timesharing terminal did a decade ago. Several logistics models have been implemented on microcomputers but these models were scaled down versions. These undertakings have also failed to produce documentation as to the problems encountered in using microcomputers. Why did the models need to be scaled down? What was the impact of the microcomputer's memory capacity? What hardware limitations were encountered? What software problems existed? We attempted to get answers for some of these questions before we began modifying the Reliability Maintainability Cost Model; hence, this chapter.

##### What is the Question

The last decade has seen a unprecedented growth in microcomputer technology. This growth has transformed microcomputers from toys into productive, useful tools. This growth has also led to the "gradual disappearance of demarcation lines between mainframe computers and microcomputers [36:256]" and has fueled the debate on whether or not microcomputers will replace mainframe computers. Heally suggests that this is the wrong question to ask: "technological revolutions are with us already; computing, however, is about solving problems and providing services, not just about technology [19:120]."

If we focus on solving problems, an entire new set of questions can be raised:

1. Why use microcomputers instead of mainframe computers?
2. What type of problems should be solved on microcomputers?
3. What are the limitations encountered when using microcomputers?

In answering the first question, we find there are several advantages microcomputers offer over the use of mainframe computers:

- "Cost is perhaps one of the strongest arguments for using a microcomputer to do engineering and scientific problem solving [38:7]." This incentive will continue to increase as microcomputers' prices fall due to advancing technology. The cost per byte of memory capacity for a microcomputer is five cents and for a mainframe computer is forty cents. A cost comparison of processing speed per dollar is even more dramatic:

The CPU processing time for a large computer is about an order of magnitude faster than that for a microcomputer, whereas the overall purchase price is three orders of magnitude greater [38:7].

Miscellaneous costs of running a microcomputer are also less than those of operating a mainframe computer. For example, support equipment (such as special air conditioning units) is not needed as microcomputers function in almost any reasonable environment.

Since microcomputers are connected to CRT screens or to slow-speed printers, it would seem prudent to restrict their use to problems requiring moderate amounts of overall processing time and moderate amounts of output [38:71].

We believe that these opinions are not an accurate representation of the capabilities of a microcomputer. Baer offers a flexible opinion concerning the capabilities of microcomputers:

. . . if computations are to be performed primarily on 16-bit or larger entities micros should not be selected. This might be wrong by the time this book is published but then the same argument could be repeated by changing 16 to 32 [2:435].

Time, as alluded to by Baer, has allowed technology to create a variety of microcomputers that are capable of handling programs that once were feasible to run only on a mainframe computer. Current microprocessor technology has produced five classes of computers (36:257):

1. Microcontrollers: 4- to 8-bit processors used in process control applications, toys, and calculators. They are capable of handling only small, well-defined tasks.

2. Microcomputers: more versatile than microcontrollers. It is in this class that one finds the personal computers such as the VIC-20, Commodore 64, and TRS-80 Model 4. Based on 8-bit processors (for example, 8080, Z80, and M6800), these machines are suited for use as word processors, smart terminals, and educational tools. The processors of this class have a memory addressability of only 64k bytes, which limits the ability of these machines to handle large

and more complex problems.

3. Minimicrocomputers: use the 8088, 8086, and 80186 family of processors. These processors are capable of performing  $10^5$  to  $10^6$  operations per second and have the ability to address 1 megabyte (1 million bytes) of memory. The IBM-PC and TRS-80 Model 2 computers belong to this class of microprocessors.

4. Maximicrocomputers: based on processors such as the Z8000, MC68000, and the 80286.

5. Supermicrocomputers: use the iAPX 432 or NS16032 processors.

The hardware reasons behind shunning microprocessors for use in large applications have been focused in the following areas (38:9):

- processor speed,
- memory addressability,
- accuracy and speed of numeric computations and,
- input/output capabilities.

The last three classes of microprocessors overcome these barriers. The remainder of this section focuses on these barriers in relation to the last three classes of microprocessors. Also note the term microcomputer will be used to represent a microprocessor from any of the five classes.

Processor Speed. Consider the microcomputer configuration set forth in Chapter 1, which is a member of class 3. The IBM Personal Computer uses an Intel 8088 microprocessor.

In terms of raw computational capability the IBM-PC has the ability to outperform by a wide margin the previous generation of personal computers. A computer graphics benchmark which scales 16,384 pairs of 16 bit integers by a fractional scale factor runs ten times faster on a 5 MHz 8008 than on a 6 MHz Z80B [17:134].

This speed comparison is in relation to a microcomputer from class 2. Turning around and looking at how the IBM-PC compares to mini and mainframe computers we can see that the implementation of large processor bound applications becomes feasible in this class of microcomputers. Table II displays the results of an IBM-PC versus various minicomputers using the Whetstone test. This test was developed by Wichman and Curnow and performs integer and real number calculations in a variety of loops and subroutine calls (43:48).

TABLE II

IBM-PC versus Minicomputer Comparison (43:49)

<u>Machine</u>	<u>Speed</u> <u>1000 Whetstone</u> <u>Instr/sec</u>
IBM-PC + 8087	72.8
PDP 11/34	181.0
PDP 11/44	252.9
VAX 11/780	668.3

In a test performed by BYTE magazine CALPASS3, a program used to predict energy use in conventional and passive solar buildings was converted to run on an IBM-PC. Runtime for CALPASS3 approaches one minute on a large mainframe compared

. . . demonstrates that it is possible to integrate the full functionality of a high-end 32-bit minicomputer or mainframe computer onto a small number of chips in a conservative NMOS process and achieve a performance-to-price ratio which compares quite favorably with such systems [21:557].

An attractive performance-to-price ratio is achievable. Stritek, Inc., of Cleveland has developed processor circuit cards utilizing the 16032 or 80286 processors that plug into an IBM-PC which effectively converts it from a class 3 microcomputer to a class 5 microcomputer for a cost of less than \$2500 (12; 22).

Memory Addressability. It may be the increase in microprocessor speed that makes running a large program practical on a microcomputer, but it is the increase in the processor's memory addressability that makes it possible. The 8088, 8086, and the 80186 processors are capable of addressing 1M bytes of memory. This is 16 times the memory capacity that a processor of class 2 can address. This fact becomes even more remarkable when we consider that the IBM 360/30 has only 65K bytes of memory and the IBM 1401 had only 12K bytes of memory. Processors of class 4 and 5 are even more powerful supporting virtual memory implementations.

Virtual memory allows you to combine a minimum of expensive primary storage (main memory) with lower-cost secondary memory. In this way, you can take full advantage of extremely large operating system software and applications programs (now offered on large mainframes) without worrying about the hardware limitations of your systems [25:53].

Table III displays the memory capacities for various processors.

TABLE III  
Microprocessors Memory Capacities  
In Bytes (25; 45)

<u>Microprocessor</u>	<u>Real Memory</u>	<u>Virtual Memory</u>
8088	1M	---
8086	1M	---
80186	1M	---
80286	16M	16
16032	8M	16M
iAPX 432	16M	1024G

It should be apparent that the memory capacity of a microcomputer no longer presents a barrier to using the microcomputer for large applications.

Numeric Capability. Numeric computational accuracy is another concern in computer applications. A typical mainframe computer has a 32-bit word (4 bytes) which is used to store a single precision floating-point number. Double precision numbers require 64-bits. Mainframe computers also have special arithmetic processors that efficiently process floating-point calculations.

Earlier microcomputer architecture did not lend itself to numerical applications for two reasons: memory capacity and numeric processing speed. Applications requiring large arrays could not fit into memory. For example, if you needed to multiply two 100 by 100 single precision matrices together,

array storage would require 80,000 bytes. Even if the arrays were small enough to fit into memory, floating-point calculations were performed via software emulation which created an overhead that effectively nullified any significant numerical analysis programs from being implemented.

As mentioned, class 3, 4, and 5 microcomputers have overcome the memory barrier. They also have overcome the speed problem of numeric processing with the development of special numerical data processors (NDP) which are specifically designed for high performance numeric processing.

Intel's 8087 NDP is one of these processors. The 8087 NDP acts as a coprocessor performing all floating-point calculations. The 8087 NDP also frees the programmer from worry about accuracy as all computations are performed using 80-bits, allowing the 8087 NDP to represent numbers maintaining 64-bits of accuracy with powers as large as  $10^{4932}$ . Few applications will exceed this capability. The 8087 NDP is also efficient. Table IV gives comparisons of the speed of selected instructions of the 8087 NDP compared to software emulation using an 8086.

TABLE IV

**Floating-Point Execution Speed  
In Microseconds (45:3-181)**

<u>Instruction</u>	<u>8087 NDP</u>	<u>8086 Emulation</u>
Add/Subtract	17	1,500
Multiply (single precision)	19	1,600
Multiply (extended precision)	27	2,100
Divide	39	3,200
Compare	9	1,300
Square Root	36	19,600
Tangent	90	13,000
Exponentiation	100	17,100

The 8087 NDP is not the only numeric data processor available. The 80287 and 16081 NDPs are used in conjunction with the 80286 and the 16032, respectively.

Input/Output. The final hurdle of implementing large programs on microcomputers deals with input/output (IO) capabilities. IO is discussed from two viewpoints: the data bus and the physical IO devices.

The data bus links together the microprocessor, memory, and IO devices and establishes the maximum data transfer rate between components. Faster and larger mass storage devices have been developed, but one should not expect to improve the performance of a program by simply attaching these faster

devices to the computer. An example will serve to demonstrate this idea. The Winchester disk used in the IBM-XT is capable of transferring data at a speed of 625k bytes per second (46:1-196). The data bus of the IBM-XT is capable of handling data transfer at a maximum of only 350K bytes per second; therefore, attaching a disk drive capable of even faster speeds accomplishes nothing. In reality we even fail to drive the data bus at its maximum rate because the disk controller operates at speeds slower than the bus. What we eventually achieve is an average data transfer rate between 60K to 90K bytes per second (9:307).

There are several ways to increase this average transfer rate. First, a faster disk controller can be produced but this leaves us bound by the maximum speed of the data bus. Second, if a faster controller is produced, one could then make use of the faster data buses found in class 4 and 5 microcomputers. Third, software techniques of caching\* and spooling can be implemented but again we are bound by the maximum rate of the data bus.

Each of the previously mentioned solutions would increase the data transfer rate but does not allow for multiple simultaneous data transfers as found on mainframe computers.

---

\*A software program which maintains blocks of frequently used disk data in memory so they will be readily available for processing (12:56).

1. operating systems,
2. supporting language processors, and
3. the application programs.

The last aspect will not be discussed here but is covered in the next chapter dealing with program implementation.

Operating Systems. It is the responsibility of the operating system to perform IO, maintain file structures, and manage memory allocation. Given these tasks, operating systems "often determine the ultimate potential of a computer system [33:188]." Only recently have operating systems for microcomputers become sufficiently sophisticated to properly handle the hardware at their disposal. Slow-speed devices are now being spooled, which makes more efficient use of the processor. Caching techniques have been implemented to achieve greater data transfer rates to and from mass storage devices. Hierarchical files structures are also allowed by several operating systems.

Memory management has also taken a giant step forward with the implementation of memory overlay structures. For class 3 microcomputers this was an important step since microprocessors in this class do not support virtual memory.

Language Processors. As with operating systems, language processors have also improved. Several years ago the only high-level language available on a microcomputer was BASIC. Today, one has the choice of FORTRAN, Pascal, C, LISP, and APL. The availability of these language processors has

## V. Implementation of Two Logistics Models

### Overview

This chapter documents the events that took place during the modification of the Reliability Maintainability Cost Model (RMCN) and the Network Repair Level Analysis (NRLA) model to run on the microcomputer described in Chapter 1. The events identified during the conversion process by no means represent an all-encompassing experience but do serve as a starting point for future endeavors.

### Model Selection

We began in December 1983 trying to locate source code and documentation for logistics models dealing with product design. We first turned to the DLSIE catalogues of logistics models. These catalogues are published on a quarterly basis and contain the descriptions of newly developed and in-work logistics models. Each catalogue covers several categories of models (for example, personnel, training, and maintenance). We identified several potential candidates and ordered additional documentation. Unfortunately, the documentation provided was only a more detailed description of the model including the assumptions and mathematical formulations used. Computer source code was not provided. Calling the contacts listed in the DLSIE catalogues we found we could obtain source code listings for several of the models while source code listings for other models were held under

proprietary rights.

We continued our search by talking to instructors in the School of Systems and Logistics and personnel at the Human Resources Laboratory as to what logistics models were being used at Wright-Patterson AFB. It was through these conversations that the RMCM and the NRLA models were identified to us as not only being suitable for testing the capacity of a microcomputer to handle large logistics models but also could be used to validate the ability of our decision function in selecting an appropriate model. We also received the added benefit that computer source code for both models were available. Source code for the RMCM was located on the CDC Cyber 6600 mainframe computer and the source code for the NRLA model was located on the Harris 500 mainframe computer.

#### Obtaining the Source Code

Even though the source code was available, we could not access it because we lack accounts for both mainframe computers. Though it was not particularly difficult to get an account as AFIT students, it still took us a week before all the paperwork was completed.

After obtaining account numbers, we discovered we still did not have authorization to read the source files as they were protected by passwords. This required us to contact the owners of the files for permission and luckily they were kind enough to allow us access.

With access allowed, we opted to use telecommunications

to download both models to the microcomputer. This eliminated the problem of entering the source code by hand, which is a tedious process and prone to errors. Although telecommunications saved us the undesirable task of entering the code by hand, telecommunications is not without its drawbacks:

1. The microcomputer user must have an account on the mainframe computer.
2. The microcomputer and mainframe computer must have telecommunications capabilities.
3. Data transfers at 300 baud are slow. Even at the faster speed of 1200 baud, data transfers of large programs can be quite time-consuming. The RMCM model's file size was approximately 376K bytes and the data transfer took almost 45 minutes.
4. Data transfers using telecommunications are not always error-free. In the transfers of both models we were fortunate in that the errors generated were obvious as they were of the garbled data type versus dropped characters.

#### What Do They Do?

With the source code for both models downloaded to the microcomputer, we turned our attention to understanding just how the programmers made the models do what they were supposed to do. We started with the RMCM. Sitting down with the listing of the program and the documentation manuals, we began to trace line-by-line through the program to understand how the program worked. This was quite time-consuming

and it was at this juncture in our research that we realized that we would not have enough time to process the NRLA model in a like manner. Although, it took approximately four weeks to complete this process, it was time well spent because during this process, we discovered where the major thrust of the conversion process would focus -- the translation of character string variables.

It should also be noted that this process was facilitated by the abundance of comment cards found in the program listing and by the well-written documentation manuals provided.

With a basic understanding of the program, we began the process of adapting the program to the microcomputer.

#### Source Code Conversion

The particular version of the RMCM we obtained was written in CDC FORTRAN IV Extended. The following obstacles were encountered in the conversion process:

Editing Capabilities. The line and screen editors available for the IBM-PC could not handle editing the complete source file of RMCM; therefore, it was necessary to break the program into smaller source code segments. This proved to be a nuisance in that it limited the use of editing capabilities such as global search/replace and block copy.

Nonstandard Statements. CDC FORTRAN IV Extended provides for several statements not implemented in FORTRAN 77 and uses different syntax structure for some statements found in FORTRAN 77. The PROGRAM and OVERLAY statements in CDC

FORTTRAN IV Extended have no counterpart in FORTRAN 77 but luckily could be disregarded during the conversion process. SUBROUTINE and CALL statements allowing for alternate returns required modification due to differences in syntax. ENCODE and DECODE statements allowed by CDC FORTRAN IV Extended required conversion to READ and WRITE statements using FORTRAN 77 internal file capabilities.

Character Data. The use of character data was the most difficult problem to overcome during the conversion process. This problem stemmed directly from the 64-bit word of the CDC Cyber 6600, which is capable of storing 10 characters per word. The IBM-PC is able to store only four characters per word. The use of the CHARACTER statement, which allows variable length character strings, provided in FORTRAN 77 facilitated the conversion but in turn created a waste of storage and hindered run time efficiency.

#### Program Debugging

With several FORTRAN 77 compilers available for the IBM-PC, it was necessary to decide which compiler we would use in the conversion process. Not all FORTRAN compilers support a complete implementation of FORTRAN 77; for instance, Supersoft FORTRAN produced by Small Systems Services, Incorporated, allows for only 16-bit integers and does not support input/output statements defined by FORTRAN 77. We selected Microsoft's implementation as being the closest implementation of FORTRAN 77 we could find.

With the source code modified to FORTRAN 77 it was time to begin the process of debugging the modified version of the model. The first step was the elimination of the syntax errors created during the conversion process. It was during this process that the capabilities of Microsoft's FORTRAN 77 Version 3.1 proved to be a nuisance. The compiler was unable to handle large source files and frequently aborted giving the message "internal compiler error." To avoid this problem it was necessary to further break the source files into smaller segments. Since separate compilations were required, the ability of the compiler to check for differences in common block sizes and improper argument types for subroutines and functions was nullified. The overall process of just removing syntax errors took approximately four days.

With the syntax errors removed we were ready to begin operational testing of the converted model. Testing was greatly facilitated by the presence of example data files and outputs in the documentation manuals. Besides the obvious errors of spelling variable names incorrectly and misunderstanding the logic, we encountered three significant problem areas:

1. Support provided by CDC FORTRAN IV Extended for disk files differs from support provided by FORTRAN 77. When opening a nonexistent disk file for output in FORTRAN 77, the file attribute must be specified as NEW; therefore, it was necessary to extensively modify the subroutine CHECK to handle

this situation. The program kept crashing until we identified all the places from which disk files were being opened, and modified the calls to subroutine CHECK.

The End-Of-File function also works differently. The logic in the program worked under CDC FORTRAN IV Extended, but it was necessary to modify the program to handle this difference. To alleviate the problem the end-of-file detection capability allowed in READ statements was used.

2. The numeric accuracy of the CDC Cyber 6600 differs from the numeric accuracy of the IBM-PC. Using a single precision variable, the Cyber carries fifteen significant digits compared to seven significant digits for the IBM-PC. When using a microcomputer, this problem can be overcome by performing all computations in double precision. Double precision variables on a 32-bit computer allow fifteen digits of accuracy to be carried. Double precision usage is not a total panacea in that it doubles core storage requirements and reduces computational speed.

Table V shows the difference in accuracy using single and double precision calculations. Since data normally consists of only estimates during the conceptual design phase, the user should note that computations using single precision calculations may be acceptable.

TABLE V

## Comparison of Single and Double Precision Calculations

<u>Computational Item</u>	<u>Single</u>	<u>Double</u>	<u>Percent Change</u>
1. Life Cycle Cost	69,951,620	69,985,582	.04%
2. Recurring Cost	37,591,720	37,625,675	.09%
3. Annual Recurring Cost	2,506,115	2,508,378	.09%
4. Non-recurring Cost	32,359,910	32,359,907	.00%

3. Microsoft's FORTRAN 77 Version 3.1 uses the IEEE standard for representing floating point numbers. We noted this situation only because the program creates two binary files to be used by a separate report generator program. This is of no concern as long as language processors supporting this standard are used, but it is a user-beware situation. For instance, Microsoft's Basic Interpreter uses a different real number representation than the IEEE standard, which effectively prohibits passing binary files between programs produced by these language processors.

The use of binary files also made it difficult to determine if the correct data was being written. Reading a hexadecimal data dump is quite tedious. To aid in debugging and program compatibility, the program was modified to support extra files. These extra files are the ASCII representation of the binary data files.

With these three problems identified, the debugging process became an iterative process of running the model,

noting errors, determining the logic fix, modifying the source code, and recompiling the program. The debugging phase took the better part of three weeks.

#### Running the Model

No changes were made in the operational commands of the model. The commands operate as documented in the RCMC User's Guide. Input data file structure also remains the same. Procedures for starting the model have been modified and reflect operation of the model on the IBM-PC using DOS 2.0 for its operating system. Operating procedures are found in Appendix D. Appendix E has examples of the model's inputs and outputs as computed when run on the IBM-PC.

#### Summary

The above discussion represents a summary of the activities that took place in converting the RCMC from the CDC Cyber 6600 to run on the IBM-PC. The occurrences of the problems mentioned were often random and not readily obvious but took more time than anticipated to overcome; therefore, we were unable to modify the NRLA model to run on the IBM-PC.

## VI. Conclusions and Recommendations

### Conclusions

In this research we have attempted to develop a methodology to identify logistics models suitable for use in a CAD environment and demonstrate the feasibility of using a microcomputer to run the selected logistics models.

Logistics Models. In the process of identifying the two logistics models to adapt to a microcomputer, we noted the following:

1. Much confusion exists as to what models are actually useful to the logistics engineer during the conceptual design phase. It was this confusion that led to the general selection function developed in Chapter 3. The function can be forced to select any particular model if the weights are assigned with bias; but if the function is used properly, it forces the logistics engineer to consider aspects of the model that could easily be ignored in a hasty decision process.

2. Current logistics models are not designed to take advantage of a microcomputer's graphics capability; therefore, it might be best to develop models from scratch versus converting them from mainframe computers. This parallels the thought of one logistics engineer who read a draft of this report. He commented that in his experience the most useful models were the simple, small ones written for a specific purpose by the engineers working on the design.

3. Logistics models are used primarily due to contractual requirements, but the model's use is only an after-thought and not normally used in the decision making process concerning a system's design. Most of the individuals interviewed believed that if the models were readily available, easy to use, and produced easy to understand outputs this attitude towards use of logistics models would change.

Microcomputers. The conversion of the RMCM demonstrates that microcomputers are capable of handling the task of running logistics models. The following conclusions are supported by the RMCM conversion.

1. Run time for the microcomputer version of the model is not prohibitive. The run time may be greater for other logistics models but with faster microcomputers becoming available run time should not be a major problem.

The model provided reasonable response times in all but one situation. This was the use of the model's GLOSSARY function. The GLOSSARY function searches a sequential file to provide the model's user definitions of key terms. The search times of this sequential file using various disk mediums are listed in Table VI.

TABLE VI  
Glossary File Search Times

<u>Type of Disk</u>	<u>Search Time (Last Entry)</u>
1. Floppy Disk	63 seconds
2. Hard Disk	47 seconds
3. RAM Disk	47 seconds

Since the search times are the same for the hard disk and the RAM disk, one can deduce that the microcomputer is processor bound and not input/output bound. A faster processor would of course improve these times. Another way to improve these times would be to rewrite the search algorithm using a random access disk file instead of a sequential file.

2. The memory capacity of the microcomputer described in Chapter 1 was more than sufficient to run the RCM. The RCM required 238K bytes of core storage to run without overlays. With the price of RAM falling with technical advances, memory capacity is no longer a limiting factor.

3. Numerical accuracy was not a significant problem (see Table V, p. 54). Conversion of models from computers having word sizes greater than 32-bits can compensate for round-off error by making use of double precision variables. The increase in run time created by the use of double precision variables on a microcomputer having special numeric processors is minimal since these processors normally perform all calculations as if the operands are double precision.

4. Software support for language processors still needs improvement. Compilers must become more reliable and generate optimum code to insure the microcomputer is used efficiently. Graphics capability for many languages is non-standard or nonexistent. FORTRAN 77 provides no direct method for using the graphics capabilities of microcomputers; therefore, it is necessary to either purchase or produce assembly language subroutines to perform graphics. Once this is done, portability of the program, which is a primary reason for using FORTRAN 77, is lost.

We also found there are several user aspects of a micro-computer versus a mainframe computer.

1. Software may not be readily convertible from one machine to another. For instance, even though the RMCM model was written in FORTRAN IV it contained several statements which were peculiar to the CDC Cyber. Thus, to enable the model to run on another mainframe which utilized FORTRAN IV, conversion of the special statements would first be necessary. When changing from mainframe to microcomputer, not only might the computer language be different but the word size might be different thus creating a problem.

2. Conversion does offer at least one advantage to the user: the chance to redesign the algorithm of the model. Due to the speed of the mainframe computer, efficient programming is not always of particular concern to the programmer. The slower speed of the microcomputer makes efficiency in

programming necessary.

3. Availability of the model to the user of a microcomputer is an advantage. A dedicated machine, even though slower than a mainframe, can provide easy access to the analysis model. Some of the individuals interviewed remarked that the difficulty of getting an account number and password for a mainframe computer, and the problems of signing on to the system were enough of a nuisance to make the use of a mainframe computer undesirable.

### Recommendations

Having completed our research we believe the following four-stage plan should be followed:

1. Using the selection function developed in Chapter 3, a study should be undertaken to identify and rank current logistics models as to the suitability for performing analysis at the conceptual design stage of a system and the model's suitability for incorporation into a CAD system.

2. The models identified should be adapted to a microcomputer. A commitment should be made at this time to the type of microcomputer system to be used so that the model's output could be adapted to graphical form. This commitment would avoid a rewrite of the graphical routines because of a change in microcomputer selection.

3. These models and microcomputers should be made available to logistics engineers to obtain their feedback

and inputs for possible modifications and enhancements.

4. The last major step will be the actual interfacing of these microcomputers to a CAD system. This would include the problems of extracting data from the CAD data base and communicating with the CAD system.

Appendix A: Logistician Interview  
(13; 15; 27; 28; 42)

This interview is part of a two-man thesis project examining computerized logistics models which would be useful during the design of a system. It is our intention to choose a model and implement it on a microcomputer. This is the first step of a larger project to integrate logistics models into a computer aided design (CAD) system. Your responses will be used to supplement our literature review.

1. For what agency do you work?

- HQ AFLC/XRS
- HQ AFLC/PTA
- HQ AFLC/MMAQP
- AFHRL/LR
- AFHRL/LRG

2. How many years have you worked with logistics models?

13, 15, 25, 27, and 30 years

3. Are you or have you been involved in the design of a system? If so, how many years?

One person definitely said no; the others were at least familiar with the process.

4. In the design stage, what types of analysis does the logistics engineer want to perform?

Life cycle cost in dollars allows the engineer to evaluate different designs in terms of dollars. Mobility and support analysis indicates the impact of the design in terms of mobility; it can look for things such as special equipment or facilities which would be required to support the design. Testability analysis is useful but is a very large and complex area. Reliability analysis is also useful.

5. Which logistics models or what types of logistics models would you consider useful in the early stages of the design of a system?

One person suggested Network Repair Level Analysis (NRLA), Logistic Support Cost (LSC), Life Cycle Cost (LCC), and reliability models. Another person argued that NRLA requires too much information to be useful during the early design stages and that NRLA is not valid for use during the early design stages. Still another person said that all

models dealing with early design that he knew of had already been put on microcomputers. Lastly, one person said that analysis of conceptual design is not done at the CAD stage. The few models that are applicable in the conceptual design phase are not adaptable to a microcomputer. This person felt that existing models should not be used with CAD systems because the process will be done differently to take advantage of CAD capabilities.

6. What models do you use in your work?

NRLA, LSC, LCC, and reliability models.

7. In what order would you rank these models if ranking them for their usefulness and importance? Why?

Only one person would rank the models and he did so using ease of program adaptability to a microcomputer as the ranking measure. He listed LSC, NRLA, and LCC2-A going from easiest to hardest.

8. What are the problems encountered in using these models (for example, data availability, data validity)?

Data availability and format were identified as problems by one individual.

9. Would being able to run these models earlier in the design process be of value?

One person stated, "Probably" but the others either stated they did not know or did not answer the question.

10. Logistics models will probably run slower on a microcomputer than on a mainframe computer. What percentage decrease in speed would be acceptable to you considering you would probably have greater access to the microcomputer?

No direct response was received for this question. The respondents believed for the most part that they were not the ones who could answer the question and some stated that they were not qualified to answer it.

11. How much of the models which you work with do you use? That is, do you usually use 50% of the capacity of the model; do you usually use 50% of the capability of the model?

No one responded to this question.

12. Of the models you work with, which features do you consider significant and which features do you believe you

could do without in order to have the model placed on a microcomputer?

All felt that this question could not be answered adequately since the answer changes by the type of analysis needed to be performed for any given project.

13. Do you have any experience with microcomputers? If so, what? Do you know of logistics models which might be adapted to microcomputers?

None had enough experience that they felt they could respond with any authority.

14. Other comments (listed in a random fashion):

- Try streamlining the program by reducing the number of inputs required. Use military standards for many of the inputs.

- The use of CAD graphics capability is more helpful than just printing out a table of numbers. An example would be stress analysis or thermal analysis in which the output is in terms of colors.

- The design engineer does not do conceptual analysis on his design; he gets feedback from the logistics engineer and the logistics planner. This analysis comes after he has made his design.

Appendix B: Computer Specialist Interview  
(10; 16; 18; 31)

This interview is part of a two-man thesis project examining computerized logistics models which would be useful during the design of a system. It is our intention to choose a model and implement it on a microcomputer. This is the first step of a larger project to integrate logistics models into a computer aided design (CAD) system. Your responses will be used to supplement our literature review.

1. How many years have worked with computers/microcomputers?

Computers	Microcomputers
8	5
10	6
16	6
17	7

2. With how many types of microcomputers are you familiar (for example, IBM, Apple, TRS-80, Commodore)?

All worked on at least one type of microcomputer, most worked on more (Apple, Cromemco, and Z-100 were mentioned).

3. What languages can you program in (for example, BASIC, FORTRAN, Pascal, Assembly)?

All knew at least BASIC with most knowing another such as FORTRAN or Assembly language.

4. Do you use microcomputers at work? At home?

All used them at least at work.

5. What are some of the problems you see, given the current structure and architecture of microcomputers, in implementing a large program (for example, a logistics model) on a microcomputer? What are your suggested solutions, if any, to these problems? Please address the following areas and any other you consider significant.

- a. Processor speed
- b. Memory considerations
- c. Input/output channels (speed and number of channels)
- d. Operating systems
- e. Programming languages

One response was that concern with processor speed was valid but that it washes out in the tradeoff; that is, one is usually willing to give up some speed to have a dedicated machine. This person felt that the types of problems that would be answered by a microcomputer were not the ones which needed to be answered in the next 15 seconds; rather, the type of applications for the micro are those for which the problem is presented in the morning and the answer is needed that afternoon. This person felt that the real use of the micro-computer was in supporting some type of decision process.

Another person believed that speed was all-important, that faster feedback was what one should be after. This person also stated that he would not put a large program on a microcomputer because it would serve no useful purpose. He defined a microcomputer as a machine similar to an Apple II. He felt that what really should be done with microcomputers is to use them in decision support systems.

Memory capacity is no longer a limiting factor in the use of microcomputers.

Problem set-up and input/output are the biggest constraints.

6. Do you see the 16/32 bit microprocessors eliminating any of these problems?

Yes. Most felt that 16-bit processors are eliminating problems with 8-bit processors and that 32-bit processors will eliminate those problems with 16-bit processors.

7. What do you see as the minimum configuration of a microcomputer required to run a large program?

Most felt that this was a hard question to answer because of the trouble of determining what is a minimum configuration and what is a large program. Together, the answers would seem to favor a microcomputer with at least 64k memory, a five megabyte hard disk, and as fast a processor as possible. One told of a program he adapted to a 64k machine and how he had to write and rewrite the code to be able to get it to fit into the 64k memory. He felt that 64k was probably too small but if a minimum was to be set it would be 64k. A hard disk was mentioned as a requirement due to the amount of data required to processed by a logistics model.

8. Other comments made were (listed in random fashion):

- Avoid describing what is inside the box but talk in

terms of inputs and outputs. It should be user friendly (even at the expense of speed or memory), easy to get into and out of the program, able to add data or to enter data easily (that is, it should use global inputs), easy to trap input errors, and able to selectively choose items to be included in the output.

- The poorest user of the system should not be able to crash the system.

- Graphics output would be useful.

- Use commercial software as much as possible due to the power and quality of the software.

- Consider doing beta testing on your software; let some classmates, both with and without computer experience, use it to observe the types of problems they have.

- The final version of your program will probably be much slower than you had anticipated.

- Evaluate what the user is going to have to do to operate the model. If it is complicated, it would not be used very much, if at all. Keep it as simple as possible for the user.

- While the machine is number crunching, consider keeping a prompt on the screen to let the user know the machine is still operating and not hung-up.

- The key in using a microcomputer is to use it for limited objectives; do not expect it to be a mainframe.

- Improve the graphics capability and the presentation of the output.

Appendix C: Description of Models  
(6; 41)

Model Name: Network Repair Level Analysis Model (NRLA)

Applicable Life Cycle Phase:

1. Development phase
2. Detail Design phase
3. Production phase
4. Operation Phase

Function: Computes life cycle costs associated with various levels of repair.

Solution Technique: Network analysis

Input Requirements:

1. Weapon system data
  - a. Number of bases
  - b. Number of operating hours
2. Maintenance system data
  - a. Labor rates
  - b. Other factors
3. Supply system data
4. Support equipment data
  - a. Cost
  - b. Availability
5. LRU data
6. SRU failure

Input Sensitivity: Estimated data may be used but one should remember that "cost for support equipment acquisition and maintenance is often critical for determining repair levels which minimize total costs [h2:61]."

Outputs:

1. Repair level decisions
2. Detailed costs
3. Cost sensitivity analysis

Implementation/Cad Data:

Language: FORTRAN IV

Run Time: unknown

**Program Size:**

**Source Code:** 4400

**Core Requirements:** unknown

**Special Peripherals:** None

**Execution:** Batch

**Graphic Output:** None, but the model is designed to accomplish sensitivity analysis which could be presented in graphical form.

**CAD Data Base Interaction:** LRU data could be extracted and then matched against a common data base.

Model Name: Reliability Maintainability Cost Model (RMCM)

Applicable Life Cycle Phase:

1. Conceptual phase
2. Development phase
3. Detail desing phase
4. Production phase
5. Operation phase

Function:

1. Computes reliability and maintainability parameters of a weapon system.
2. Computes life cycle cost of a system.

Solution Technique:

1. Probability theory.
2. Accounting theory.

Input Requirements:

1. Frequency of maintenance actions.
2. Task/event data with each maintenance action:
  - a. Type
  - b. Probability
  - c. Avworage time to complete
  - d. Manpower and skill requirements
3. Cost elements

Input Sensitivity: can use estimated data

Outputs:

1. Man hour resource requirements
2. Reliability parameters
3. Maintainability parameters
4. Availability parameters
5. Cost of system

Implementation/CAD Data:

Language: CDC FORTRAN IV Extended

Run Time: unkown

Program Size:

Source Code: 4400

Core Requirements: unknown

**Special Peripherals: None**

**Execution: Interactive**

**Graphic Output: None, but the model is designed to accomplish sensitivity analysis which could be presented in graphical form.**

**CAD Data Base Interaction: LRU data could be extracted and then matched against a common data base.**

## Appendix D: Disk File Setup

The following information details the requirements for initializing the data files prior to running the RMCM on the IBM-PC under the Disk Operating System (DOS) 2.0. It is assumed that the reader is familiar with DOS 2.0 and its file structure. For information on DOS 2.0 reference IBM's publication #6024061: Disk Operating System.

### Disk File Setup

All data disks files must be located on the "C" disk drive. The RMCM program module may be located on any available drive. The following data files are mandatory and must be placed in the current working directory prior to starting the RMCM:

1. RMBASE: reliability data file
2. COST: cost input data file.

The following data files are optional and must be included in the current working directory only if the indicated function is to be accessed by the user:

1. HELP: contains helpful messages for the user.

Used by the HELP function.

2. DEFINE: contains definitions of key terms.

Used by the GLOSSARY function.

3. RMPERT: if the user desires to use a perturbed data file from a previous session, it is necessary to copy the desired perturbed data file into the current working

directory under this name.

The following data files may be created during an execution of the RMCN:

1. TEMP-3: temporary data file used by the model.  
May be deleted at the end of the session.
2. TEMP-4: temporary data file used by the model.  
May be deleted at the end of the session.
3. TEMP-7: temporary data file used by the model.  
May be deleted at the end of the session.
4. RMPERT: if not previously included this file  
will be created if the user perturbs reliability or cost  
data.
5. BSEOUT: binary output file computed using base  
data. Used by the model and possibly subsequent report  
programs.
6. PRTOU: binary output file computed using  
perturbed data. Used by the model and possibly subsequent  
report programs. This file is created only if the user  
decides to perturb reliability or cost data.
7. BSEOUT.ASC: ASCII representation of the BSEOUT  
file.
8. PRTOU.ASC: ASCII representation of the PRTOU  
file.

Example Session One

THE RELIABILITY, MAINTAINABILITY AND COST MODEL

DO YOU WANT BASIC INSTRUCTIONS (Y OR N) ?  
N

FUNCTION?  
MODIFY

FOUND R&M FILE TO COPY.

R+M VARIABLE?  
MFHBM A

NEW TITLE?  
TEST #1

TYPE?  
FACTOR

FACTOR =  
1.2

MASK=  
AFR

DO YOU WANT A LISTING OF THE CHANGED ITEMS?  
Y

EQUIP	RMBASE	RMPERT
AFRRF	118.8	142.6
AFRDI	160.1	192.1
AFRME	696.8	836.2

3 CHANGES.

FUNCTION?  
PRODUCTS

FOUND THE R&M BASE FILE.

FOUND A PERTURBED DATA FILE.

COMPARE WITH PERTURBED R&M FILE?  
Y

SIMILARIZING PERTURBED DATA FILE.

FINISHED SIMILARIZING PERTURBED DATA FILE.

TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION RM DATA

FINISHED READING R&M DATA.

TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION RM DATATEST #1

FINISHED READING R&M DATA.

DATA TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION COST DATA

FINISHED READING COST DATA.

DO YOU WANT TO CHANGE INITIAL COST INPUTS?

N

DO YOU WANT TO PERTURB COSTS?

N

PERTURBED OUTPUT FILE TITLE?  
TEST #1

REPORT?  
LCC

DO YOU WANT:  
1 - % CHANGE  
2 - DIFFERENCE ?

1

*LCC	BSEOUT	PRTOUT	% CHANGE
82,460,720.0	78,656,010.0		-4.6

REPORT?  
RCY

DO YOU WANT:  
1 - % CHANGE  
2 - DIFFERENCE ?

1

*RCY	BSEOUT	PRTOUT	% CHANGE
3,340,055.0	3,194,000.0		-4.4

REPORT?  
END

RMCM ENDED  
Stop - Program terminated.

Example Session Two

THE RELIABILITY, MAINTAINABILITY AND COST MODEL

DO YOU WANT BASIC INSTRUCTIONS (Y OR N) ?  
N

FUNCTION?  
PRODUCTS

FOUND THE R&M BASE FILE.

FOUND A PERTURBED DATA FILE.

COMPARE WITH PERTURBED R&M FILE?  
N

TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION RM DATA

FINISHED READING R&M DATA.

DATA TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION COST DATA

FINISHED READING COST DATA.

DO YOU WANT TO CHANGE INITIAL COST INPUTS?  
N

DO YOU WANT TO PERTURB COSTS?  
N

REPORT?  
LCC  
\*LCC            BSEOUT  
                 69,951,620.0

REPORT?  
RC  
\*RC            BSEOUT  
                 37,591,720.0

REPORT?  
RCY  
\*RCY           BSEOUT  
                 2,506,115.0

REPORT?

NRC  
\*NRC

BSEOUT  
32,359,910.0

REPORT?  
END

RHCH ENDED  
Stop - Program terminated.

### Example Session Three

#### THE RELIABILITY, MAINTAINABILITY AND COST MODEL

DO YOU WANT BASIC INSTRUCTIONS (Y OR N) ?  
N

FUNCTION?  
PRODUCTS

FOUND THE R&M BASE FILE.

FOUND A PERTURBED DATA FILE.

COMPARE WITH PERTURBED R&M FILE?  
N

TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION RM DATA

FINISHED READING R&M DATA.

DATA TITLE CARD READ:  
BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION COST DATA

FINISHED READING COST DATA.

DO YOU WANT TO CHANGE INITIAL COST INPUTS?  
N

DO YOU WANT TO PERTURB COSTS?  
Y

COST VARIABLE?  
UC

TYPE?  
FACTOR

FACTOR =  
1.2

MASK=  
AFR

DO YOU WANT A LISTING OF THE CHANGED ITEMS?  
Y

	COST	PERTURBED
AFRRF1	129141.00	154969.20

AFRRF2	115398.00	138477.60
AFRD11	62321.00	74785.20
AFRD12	71556.00	85867.20
AFRME1	58529.00	70234.80
AFRME2	95265.00	114318.00

6 CHANGES.

COST VARIABLE?

X

PERTURBED OUTPUT FILE TITLE?

TEST #2

REPORT?

LCC

DO YOU WANT:

1 - % CHANGE

2 - DIFFERENCE ?

1

*LCC	BSEOUT	PRTOUT	% CHANGE
	69,951,620.0	74,681,380.0	6.8

REPORT?

NRC

DO YOU WANT:

1 - % CHANGE

2 - DIFFERENCE ?

1

*NRC	BSEOUT	PRTOUT	% CHANGE
	32,359,910.0	37,030,280.0	14.4

REPORT?

RC

DO YOU WANT:

1 - % CHANGE

2 - DIFFERENCE ?

1

*RC	BSEOUT	PRTOUT	% CHANGE
	37,591,720.0	37,651,100.0	.2

REPORT?

END

RMCH ENDED

Stop - Program terminated.

### Sample Reliability Data

BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION RM DATA  
03

CR AFRRF	-1	123.5	75A00	1	RADIO FREQUENCY SUBSYSTEM				2	
CR AFRRF1	-1	69.2	75AA0	1	TRANSMITTER LRU				8	
CR AFRRF2	-1	54.3	75AB0	1	LOW POWER RADIO FREQUENCY LRU				12	
CR AFRD1	-1	94.7	75B00	1	DIGITAL SUBSYSTEM				2	
CR AFRD11	-1	31.3	75BA0	1	COMPUTER LRU				11	
CR AFRD12	-1	63.4	75BB0	1	DIGITAL SIGNAL PROCESSOR LRU				36	
CR AFRME	-1	78.7	75C00	1	MECHANICAL SUBSYSTEM				2	
CR AFRME1	-1	61.7	75CA0	1	ANTENNA LRU				5	
CR AFRME2	-1	17.0	75CB0	1	RACK LRU				0	
CR AFRME2	-2		75CB0							
SF AFRRF	-1	PCHYD	PCHYD		HANDF	PCHYD	PCHYD	PCHYD	1	
SF AFRD1	-1	PCHYD	PCHYD		HANDF	PCHYD	PCHYD	PCHYD	1	
SF AFRME	-1	PCHYD	PCHYD		HANDF	PCHYD	PCHYD	PCHYD	1	
LF AFRRF	-1	32657	32657		32657	32657	32657	32657	1	
LF AFRD1	-1	32657	32657		32657	32657	32657	32657	1	
LF AFRME	-1	32657	32657		32657	32657	32657	32657	1	
LS AFRRF1	-1		32654	32654	32654			32654	32654	1
LS AFRRF2	-1		32654	32654	32654			32654	32654	1
LS AFRD11	-1		32654	32654	32654			32654	32654	1
LS AFRD12	-1		32654	32654	32654			32654	32654	1
LS AFRME1	-1		32654	32654	32654			32654	32654	1
LS AFRME2	-1		32654	32654	32654			32654	32654	1
TS AFRRF1	-1		13	3	13			40	40	
TS AFRRF2	-1		12	2	12			40	40	
TS AFRD11	-1		10	2	10			40	40	
TS AFRD12	-1		11	2	11			40	40	
TS AFRME1	-1		8	1	8			40	40	
TS AFRME2	-1		10	1	10			40	40	
TF AFRRF	-1	1	2	1	2	5	2	2		
TF AFRD1	-1	1	2	1	2	5	2	2		
TF AFRME	-1	1	2	1	2	5	2	2		
PF AFRRF	-1	10000	.7000	.3000	.6200	.0800	.6200	.0800		
PF AFRD1	-1	10000	.7000	.3000	.6200	.0800	.6200	.0800		
PF AFRME	-1	10000	.7000	.3000	.6200	.0800	.6200	.0800		
PS AFRRF1	-1		.067	.037	.082			1993	1993	
PS AFRRF2	-1		.217	.091	.126			1993	1993	
PS AFRD11	-1		.078	.060	.048			1993	1993	
PS AFRD12	-1		.195	.144	.095			1993	1993	
PS AFRME1	-1		.317	.085	.164			1993	1993	
PS AFRME2	-1		.051	.000	.003			1993	1993	
SS AFRRF1	-1		ATIT1	ATIT1	ATIT1	01		ATIT1	ATIT1	1
SS AFRRF2	-1		ATIT2	ATIT2	ATIT2	02		ATIT2	ATIT2	1
SS AFRD11	-1		ATIT3	ATIT3	ATIT3	03		ATIT3	ATIT3	1
SS AFRD12	-1		ATIT4	ATIT4	ATIT4	04		ATIT4	ATIT4	1
SS AFRME1	-1		ATIT5	ATIT5	ATIT5	05		ATIT5	ATIT5	1
SS AFRME2	-1		ATIT6	ATIT6	ATIT6	06		ATIT6	ATIT6	1
MF AFRRF	-1	118.8								

MF AFRDI -1 160.1  
MF AFRME -1 696.8

002

32657 32654

06

ATIT1 ATIT2 ATIT3 ATIT4 ATIT5 ATIT6

# Sample Cost Data

## BASELINE CONFIGURATION-WEAPON SYSTEM RADAR-ASSET DEMONSTRATION COST DATA

VE RECUR -1	0							
VE RECUR -2								
VE NRECUR -3	0			0				
VE NRECUR -4					0			
VI AFRRF1 -1	129141		.01	.01	36	1.84	100	500
VI AFRRF2 -1	115398		.01	.01	36	1.84	100	500
VI AFRDI1 -1	62321		.01	.01	36	1.84	100	500
VI AFRDI2 -1	71556		.01	.01	36	1.84	100	500
VI AFRME1 -1	58529		.01	.01	36	1.84	100	500
VI AFRME2 -1	95265		.01	.01	36	1.84	100	500
VI AFRRF1 -2	8.	0	8.					
VI AFRRF2 -2	12.	0	12.					
VI AFRDI1 -2	11.	0	11.					
VI AFRDI2 -2	36.	0	36.					
VI AFRME1 -2	5.	0	5.					
VI AFRME2 -2	0.	0	0.					
VJ ATIT1 -1	445297	0	.30			0	0	1.0
VJ ATIT2 -1	350319	0	.30			0	0	1.0
VJ ATIT3 -1	572268	0	.30			0	0	1.0
VJ ATIT4 -1	572268	0	.30			0	0	1.0
VJ ATIT5 -1	95486	0	.30			0	0	1.0
VJ ATIT6 -1	100	0	.30			0	0	1.0
VJ ATIT1 -2	1.0							
VJ ATIT2 -2	1.0							
VJ ATIT3 -2	1.0							
VJ ATIT4 -2	1.0							
VJ ATIT5 -2	1.0							
VJ ATIT6 -2	1.0							
VD ATIT1 -1	1	100						
VD ATIT2 -1	1	100						
VD ATIT3 -1	1	100						
VD ATIT4 -1	1	100						
VD ATIT5 -1	1	100						
VD ATIT6 -1	1	100						
VN 32657 -1	8	500	100	0		1000		.33
VN 32654 -1	8	500	100	0		1000		.33
VN 32657 -2	17.	0.0	1.0		11.70	2.28	0.0	
VN 32654 -2	17.	0.0	1.0		11.70	2.28	0.0	
VP 32PIL -1	15000							
VS SCALAR -0	0	0	2000	0	0	1981		
VS SCALAR -1	.13	.36	.53	.43	.1	.05		100
VS SCALAR -2	0	0	2000	2000	0	2000	0	.25
VS SCALAR -3	0	0		0	0		0	
VS SCALAR -4	0	0		0	0		.2	.2
VS SCALAR -5	40.91	104.20	20.20	.53	.99	1.35	0.1	5000
VS SCALAR -6		247	3512	2461	1920	.60	1	15
VS SCALAR -7	23	1	25	55	0	.5	.15	.5
VS SCALAR -8	420	0	0	704959		1914802	0	0
VS SCALAR -9	.10	.1	1		0	1	5	.09

## Appendix F: Program Listing

```

COMMON /OVER/ JABT,I02,I04A,JN,NMAX,LAST
C
INTEGER MASK,TITLE
COMMON /ALL/ NOTHER,KPR,K80,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
COMMON /HINT/ JHINT
C
CHARACTER*10 FUNC(5)
CHARACTER*1  AST(2)
C
DATA FUNC/'MODIFY','SET','GLOSSARY',
& 'PRODUCTS','END'/
DATA AST/' ','*'/
C
WRITE(*,10)
10 FORMAT(/,' THE RELIABILITY, MAINTAINABILITY AND COST MODEL ',/)
C
C CHECK FOR PRESENCE OF HELP FILE.
C
      JHINT = 0
      CALL CHECK(1,*1)
      GOTO 5
1  JHINT = -1
      WRITE(*,9000)
9000 FORMAT(' DID NOT FIND THE HELP FILE',/)
C
C INITIALIZE SYSTEM PARAMETERS.
C
      5 CALL RESET
C
C READ IN SYSTEM DATA FILES CURRENTLY AVAILABLE (LCCFILES).
C
      LAST = 0
      NOTHER=0
      IF (JHINT .NE. 0) GOTO 20
15 WRITE(*,16)
16 FORMAT(/,' DO YOU WANT BASIC INSTRUCTIONS (Y OR N) ?      ')
      CALL INP(3,0,0,0,0,2,J,*20,*15)
      IF (J .EQ. 1) CALL HELP(5)
C
C ALL PROMPTS FOLLOW THE FOLLOWING FORMAT:
C 1) PRINT PROMPT IF USER HAS NOT ANTICIPATED IT IN WHICH
C CASE NOTHER WOULD BE > 0.
C 2) CALL INP TO RECEIVE USER RESPONSE.
C 3) REPRINT THE PROMPT IF USER TYPES HELP (SECOND CONDITIONAL
C RETURN).
C 4) PROCEED BASED ON USER'S RESPONSE.
C
20 IF (NOTHER .EQ. 0) WRITE(*,30)

```

```

30 FORMAT(/, ' FUNCTION?      ')
   CALL INP(4,FUNC,0,0,5,10,J,*1000,*20)
   BOTO (40,50,60,100,1000),J
C
40 CALL MODIFY
   BOTO 20
C
50 CALL SET
   BOTO 20
C
60 CALL DEFINE
   BOTO 20
C
C FOR OUTPUT PRODUCTS, FIRST GET R+M FILE(S) FOR USE.  I02=0
C INFERS NO PERTURBATION OF R+M DATA IS TO BE EXAMINED.
C
100 I02=0
    JN=0
C
C CHECK FOR PRESENCE OF R+M BASE FILE.
C
    CALL CHECK(11,*120)
    WRITE(*,9010)
9010 FORMAT(/, ' FOUND THE R&M BASE FILE. ')
    BOTO 130
120 WRITE(*,121)
121 FORMAT(/, ' R+M FILE NOT ATTACHED. ')
    BOTO 1000
C
C CHECK FOR PRESENCE OF PERTURBED R+M FILE.
C
130 CALL CHECK(12,*153)
    WRITE(*,9020)
9020 FORMAT(/, ' FOUND A PERTURBED DATA FILE. ')
    IF (NOTHER .EQ. 0) WRITE(*,140)
140 FORMAT(/, ' COMPARE WITH PERTURBED R&M FILE?      ')
    CALL INP(3,0,0,0,0,200,JN,*20,*130)
    IF (JN .EQ. 0) BOTO 153
    I02 = 12
    WRITE(*,9030)
9030 FORMAT(/, ' SIMILARIZING PERTURBED DATA FILE. ')
152 CALL SIMILA (*1000)
C
    WRITE(*,9040)
9040 FORMAT(/, ' FINISHED SIMILARIZING PERTURBED DATA FILE. ')
153 NMAX=16
    JABT=0
    CALL RMODEL
    IF (JABT .EQ. 1) BOTO 20
    JABT=0
    CALL CMODEL
    IF (JABT .EQ. 1) BOTO 156

```

```

      NMAX=66
156 JABT=0
      CALL XOUT
      IF (JABT .EQ. 1) GOTO 1000
      GOTO 20
C
1000 WRITE(*,1010)
1010 FORMAT(///, ' RMCH ENDED' )
      STOP
      END
      SUBROUTINE FIND(CH,KODE)
C
C THIS ROUTINE IS CALLED BY INP TO CONVERT AN ALPHA CHARACTER
C TO A DIGIT OR SPECIAL CODE:
C   KODE = 1-10 FOR THE CORRESPONDING DIGIT PLUS 1
C       11 BLANK
C       12 COMMA
C       13 PERIOD (DECIMAL)
C       14 PLUS SIGN
C       15 MINUS SIGN
C       16 Y
C       17 N
C       0 NONE OF THE ABOVE
C
      CHARACTER*1 CH,TAB(17)
      DATA TAB/'0','1','2','3','4','5','6','7','8','9',
& ',','.','+', '-', 'Y','N' /
C
      DO 10 KODE=1,17
      IF (CH.EQ.TAB(KODE)) RETURN
10 CONTINUE
      KODE=0
      RETURN
      END
      SUBROUTINE HELP(N)
C
C READ FROM UNIT 1 ALL CARDS WITH THE VALUE N IN COLUMNS 1-4.
C CALLED FROM INP (AND MAIN INITIALLY) TO PROVIDE USER ASSISTANCE
C TO PROMPTS (AND BASIC INSTRUCTIONS).
C CALLED BY MAIN,INP
C
      CHARACTER*76 ARRAY
C
      COMMON /HINT/ JHINT
C
      DATA LAST/0/
C
      IF (JHINT .GE. 0) GOTO 5
      WRITE(*,3)
3 FORMAT(/, ' HELP FILE NOT ATTACHED. ' )
      RETURN
5 KOUNT=0

```

```

C
C THE FILE IS LEFT IN POSITION FROM THE LAST CALL AND REMOUND
C IF THE CURRENT N IS EARLIER, ADVANCED IF LATER IN THE FILE,
C OR PRINTED RIGHT AWAY IF WE HAPPEN TO WANT THE VERY NEXT HELP.
C
      IF (N-LAST) 10,40,20
10  REWIND 1
20  READ(1,30,END=99) LAST,ARRAY
30  FORMAT(I4,A76)
      IF (LAST .NE. N) GOTO 20
40  CALL ABORT(KOUNT,*60)
      WRITE(*,50) ARRAY
50  FORMAT(1X,A76)
60  READ(1,30,END=99) LAST,ARRAY
      IF (LAST .EQ. N) GOTO 40
      GOTO 100
99  LAST=99999
100 RETURN
      END
      SUBROUTINE DEFINE
C
C THIS ROUTINE ACCESSES THE GLOSSARY (FILE 'DEFINE').
C THE USER MAY ASK FOR A DEFINITION AF ANY TERM OR A LIST OR
C MASKED LIST OF THE AVAILABLE TERMS.
C CALLED BY MAIN,MODIF,OUTPUT,MODCST
C
      CHARACTER*1 LBBB,TERM(10)
      CHARACTER*4 LIST
      CHARACTER*10 TRM,DISP(7),OLD,SYN(7),ARRAY(7),ALL(7)
C
      CHARACTER*1 XMASK,XTITLE
      COMMON /JJF/ XMASK(10),XTITLE(10)
C
      INTEGER MASK,TITLE
      COMMON /ALL/ NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
      DATA J/0/
      DATA LIST/'LIST'/
      DATA LBBB/'L'/
C
C THE FIRST TIME THROUGH THE FILE IS ATTACHED AND BASIC
C INFORMATION PRINTED (UNLESS THE USER HAS ANTICIPATED THE
C NEXT PROMPT). NUM IS THE NUMBER OF LINES IN THAT BASIC INFO.
C
      CALL CHECK(2,*5)
      GOTO 7
5  WRITE(*,6)
6  FORMAT(/,' FILE DEFINE NOT ATTACHED.')
      RETURN
      IF (J .EQ. 1) GOTO 60
7  J=1

```

```

      IF (NOTHER .BT. 0) GOTO 75
10  READ(2,20) NUM,NS,SYN
20  FORMAT(13,12,5X,7A10)
C
C  PRINT NUM LINES OF THE GLOSSARY.
C
25  LINE=0
    DO 30 K=1,NUM
      READ(2,30) ARRAY
30  FORMAT(7A10)
      CALL ABORT(LINE,*50)
      WRITE(*,40) ARRAY
40  FORMAT(1X,7A10)
50  CONTINUE
C
60  IF (NOTHER .EQ. 0) WRITE(*,70)
70  FORMAT(/,' TERM?      ')
75  CALL INP(5,0,TERM,0,10,3,NUM,*210,*60)
    WRITE(TRM,80) TERM
80  FORMAT(10A1)
C
C  IF THE USER WANTS A LIST OF AVAILABLE TERMS, HE INPUTS 'L'
C  OR 'LIST'. WE THEN BYPASS THE BASIC INFO.
C
      IF (TRM .NE. LIST .AND. TRM .NE. LBBB) GOTO 140
      REWIND 2
      LINE=0
      READ(2,20) NUM
      DO 90 K=1,NUM
        READ(2,80)
90  CONTINUE
C
C  GET A MASK FOR THE LIST, IF DESIRED.
C
      IF (NMASK .EQ. -1) GOTO 102
      NMASK=NMASK
      GOTO 105
102  IF (NOTHER .EQ. 0) WRITE(*,104)
104  FORMAT(/,' MASK=      ')
      CALL INP(5,0,XMASK,0,10,50,NMASK,*60,*102)
105  KNT=0
      KSOME=0
C
C  READ THE HEADER CARD. ALL CONTAINS THE TERM AND ITS SYNONYMS.
C  AND SKIP THE DEFINITION
C
110  READ (2,20,END=130) NUM,NS,ALL
      DO 120 K=1,NUM
        READ(2,80)
120  CONTINUE
      IF (NMASK .EQ. 0) GOTO 123
C

```

```

C CHECK ALL THE SYNONYMS AGAINST THE MASK.
C
  DO 122 K=1,NS
  TRM=ALL(K)
  READ(TRM,80) TERM
  CALL MATCH(NMSK,XMASK,10,TERM,*122)
  BOTO 124
122 CONTINUE
  BOTO 110

C
C IF NO MASK, USE THE FIRST SYNONYM.
C STORE THIS TERM IN DISP. PRINT OUT IN GROUPS OF 7.
C
123 TRM=ALL(1)
124 KNT=KNT+1
  K SOME=1
  DISP(KNT)=TRM
  IF (KNT .LT. 7) BOTO 110
  CALL ABORT(LINE,*60)
  WRITE(*,126) DISP
126 FORMAT(7(1X,A10))
  KNT=0
  BOTO 110

C
C PRINT LAST GROUP AT END OF FILE.
C
130 REWIND 2
  IF (KNT .GT. 0) WRITE(*,126) (DISP(K),K=1,KNT)
  IF (K SOME .EQ. 0) WRITE(*,133) XMASK
133 FORMAT(1X,10A1,' NOT IN GLOSSARY.')
  BOTO 60

C
C CHECK FOR NEXT TERM.
C
135 REWIND 2
140 READ(2,20,END=135) NUM,NS,SYN

C
C SET STARTING POINT.
C CHECK AGAINST ALL SYNONYMS.
C
  OLD=SYN(1)
150 DO 160 K=1,NS
  IF (TRM .NE. SYN(K)) BOTO 160
  WRITE(*,155) TRM
155 FORMAT(' *',A10)
  BOTO 25
160 CONTINUE

C
C BYPASS DEFINITION.
C
  DO 170 K=1,NUM
  READ(2,80)

```

```

170 CONTINUE
180 READ(2,20,END=200) NUM,NS,SYN
    IF (SYN(1) .NE. OLD) GOTO 150
C
C IF WE GET BACK TO WHERE WE STARTED WITHOUT EVER FINDING THE
C DESIRED TERM, IT ISN'T IN THE GLOSSARY.
C
    WRITE(*, 190) TRM
190 FORMAT(/,1X,A10,' NOT IN GLOSSARY.')
    NOTHER=0
    REWIND 2
    GOTO 60
C
C AT END OF FILE LOOKING FOR A TERM, REWIND AND KEEP LOOKING.
C
200 REWIND 2
    GOTO 180
210 RETURN
    END
    SUBROUTINE ABORT(N,*)
C
C THIS ROUTINE IS CALLED JUST BEFORE EACH LINE OF A LIST IS
C BEING PRINTED. IF THE LINE IS NOT TO BE PRINTED, BECAUSE
C THE USER ONLY WANTS 'MAXLIN' LINES AT A TIME, OR BECAUSE
C THE USER HAS CANCELLED THE REMAINDER OF THE LIST, RETURN 1
C IS EFFECTED.
C N IS SET TO 0 BY THE CALLING PROGRAM PRIOR TO
C THE FIRST CALL.
C
    COMMON /LINES/ MAXLIN
C
    CHARACTER*1 CH,EX,BL
C
    DATA EX/'X'/
    DATA BL/' '/
C
C IF N IS SET TO ABORT FROM A PREVIOUS CALL, THE PRINT IS
C ABORTED.
C
    IF (N .EQ. -1) RETURN 1
C
C INCREMENT N AND PRINT ANOTHER LINE.
C
    IF (N .GE. MAXLIN) GOTO 1
    N=N+1
    RETURN
C
C WE HAVE REACHED THE LIMIT. RESET N TO 1 LINE. IF THE USER
C TYPES X, SET LOOP ABORT TO -1. ELSE RETURN WITH N BACK AT
C 1 TO KEEP PRINTING. FOR INVALID ENTRIES, HELP IS PROVIDED.
C
1 WRITE(*,20)

```

```

      N=1
      5 READ (*,10) CH
      10 FORMAT(A1)
      IF (CH .NE. EX) GOTO 15
      N=-1
      RETURN 1
      15 IF (CH .EQ. BL) RETURN
      WRITE(*,20)
      GOTO 5
      20 FORMAT(/, ' ENTER X TO CANCEL.  ENTER SPACE TO CONTINUE. ')
      END
      SUBROUTINE MATCH(NUM,MASK,NCOL,COLS,*)
C
C  THIS ROUTINE COMPARES MASK TO COLS.  IF MASK IS COMPLETELY
C  CONTAINED IN COLS, A NORMAL RETURN IS MADE.  IF NOT, RETURN 1.
C      NUM - LENGTH OF MASK
C      MASK - UP TO 10 CHARACTERS
C      NCOL - LENGTH OF THE CANDIDATE FIELD
C      COLS - THE CANDIDATE FIELD
C
C
C      CHARACTER*1 PER,MASK(10),COLS(10)
C
C      DATA PER/'.'/
C
C  NO MASK IMPLIES CONTAINMENT.
C
C      IF (NUM .EQ. 0) RETURN
C
C  JUP IS THE NUMBER OF POSSIBLE ALIGNMENTS.
C
C      JUP=NCOL+1-NUM
C      DO 20 K=1,JUP
C      DO 10 L=1,NUM
C
C  ACCEPT PERIODS AS HITS REGARDLESS.
C
C      IF (MASK(L) .EQ. PER) GOTO 10
C      IF (MASK(L) .NE. COLS(K+L-1)) GOTO 20
      10 CONTINUE
C
C  SATISFIED LOOP INFERS MATCH OF KTH ALIGNMENT POSITION.
C
C      RETURN
      20 CONTINUE
      RETURN 1
      END
      SUBROUTINE SET
C
C      CHARACTER*10 PARMS(12)
C
C      CHARACTER*1 XMASK,XTITLE

```

```

COMMON /JJF/ XMASK(10),XTITLE(10)
C
  INTEGER MASK,TITLE
  COMMON /ALL/ NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
  COMMON /LINES/ MAXLIN
C
  DATA PARMS/'LINES','DIFFERENCE','XCHANGE','SORT',
    & 'NOBORT','MASK','NOMASK','TITLE','NOTITLE','BACK',
    & 'LIST','NOLIST'/
C
C THIS ROUTINE ALLOWS THE USER TO SET PARAMETERS TO BE USED
C THROUGHOUT THE PROGRAM, INSTEAD OF BEING PROMPTED EACH TIME
C ONE OF THESE PARAMETERS IS NECESSARY.
C CALLED BY MAIN, OUTPUT, MODCST
C
  10 IF (NOTHER .EQ. 0) WRITE(*,20)
  20 FORMAT(/,' PARAMETER = ')
    CALL INP(4,PARMS,0,0,12,190,J,*45,*10)
    BOTO (30,50,60,70,80,90,110,120,140,150,160,170),J
C
C CHANGE MAX LINES TO ANY VALUE FROM 1 TO 9999.
C
  30 IF (NOTHER .EQ. 0) WRITE(*,40)
  40 FORMAT(/,' MAX LINES = ')
    CALL INP(1,0,0,1,9999,40,MAXLIN,*45,*30)
  45 RETURN
C
C USER WANTS ARITHMETIC DIFFERENCE BETWEEN BASE AND PERTURBED
C OUTPUT TO BE PRINTED.
C
  50 KPR=2
    RETURN
C
C USER WANTS X CHANGE BETWEEN BASE AND PERTURBED OUTPUT TO
C BE PRINTED.
C
  60 KPR=1
    RETURN
C
C USER WANTS ALL OUTPUTS SORTED.
C
  70 KSO=1
    RETURN
C
C USER WANTS NO OUTPUTS SORTED.
C
  80 KSO=0
    RETURN
C
C USER WANTS TO USE THE SAME MASK EVERY TIME ONE IS NEEDED.
C

```

D-A147 666

ADAPTING LOGISTICS MODELS TO A MICROCOMPUTER FOR  
INTERFACE WITH COMPUTER- (U) AIR FORCE INST OF TECH  
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST..

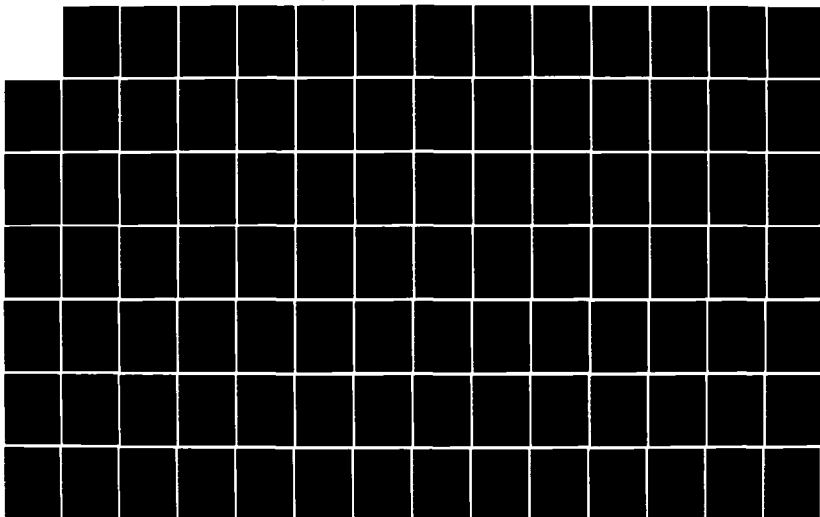
2/3

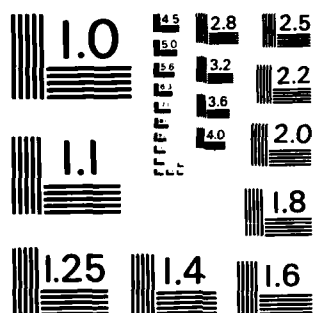
UNCLASSIFIED

D G DAVIDSON ET AL. SEP 84

F/G 15/5

NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

```

90 IF (NOTHER .EQ. 0) WRITE(*,100)
100 FORMAT(/, ' MASK=      ')
    CALL INP(5,0,XMASK,0,10,50,NMASK,*45,*90)
    RETURN
C
C  USER WANTS A PROMPT EVERY TIME A MASK IS NEEDED.
C
110 NMASK=-1
    RETURN
C
C  USER WANTS THE SAME TITLE ON ALL MODIFIED FILES.
C
120 IF (NOTHER .EQ. 0) WRITE(*,130)
130 FORMAT(/, ' TITLE=      ')
    CALL INP(5,0,XTITLE,0,10,130,NTITL,*45,*120)
    RETURN
C
C  USER WANTS A PROMPT EVERY TIME A TITLE IS NEEDED.
C
140 NTITL=-1
    RETURN
C
C  USER WANTS TO RESET PARAMETERS TO THEIR ORIGINAL VALUES.
C
150 CALL RESET
    RETURN
C
C  USER WANTS A LIST OF CHANGES WHENEVER ONE IS TO BE ASKED FOR.
C
160 KLI=1
    RETURN
C
C  USER WANTS NO CHANGES TO BE LISTED.
C
170 KLI=0
    RETURN
    END
    SUBROUTINE RESET
C
    INTEGER MASK,TITLE
    COMMON /ALL/ NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
    COMMON /LINES/ MAXLIN
C
C  THIS ROUTINE RESETS PARAMETERS IN /ALL/ TO THEIR ORIGINAL
C  VALUES, IN ORDER THAT A USER PROMPT WILL BE PRINTED EVERY
C  TIME DIFFERENCE OR % CHANGE, SORT, LIST, MASK, OR TITLE
C  IS NEEDED.  MAX LINES IS RESET TO 10.
C  CALLED BY MAIN, SET
C
    KPR=0
    KSO=-1

```

```

      KLI=-1
      NMASK=-1
      NTITL=-1
      MAXLIN=10
      RETURN
      END
      SUBROUTINE SIMILA(*)
C
      CHARACTER*10 T1,T2
C
C   THE R+M FILES USED FOR OUTPUT MUST BE SIMILAR, THAT IS,
C   CREATED FROM THE SAME ORIGINAL DATA BASE, SO AS TO HAVE
C   THE EXACT SAME NUMBER OF EQUIPMENTS, ETC. TO ASSURE THIS
C   COLUMNS 3-12 HAVE A UNIQUE CODE WHICH IS KEPT DURING
C   PERTURBATION OF FILES. IF UNEQUAL, NO DICE.
C
      REWIND 11
      REWIND 12
      READ(11,10) T1
      READ(12,10,END=99) T2
10  FORMAT(/,2X,A10)
      REWIND 11
      REWIND 12
      IF (T1.EQ. T2) RETURN
99  WRITE(*,20)
20  FORMAT(/,' SO SORRY!  -- INCOMPATIBLE FILES')
      RETURN 1
      END
      SUBROUTINE INP (JFLAG,TABLE,TAB2,MIN,MAX,JHELP,INT,*,*)
C
C   THIS SUBROUTINE IS USED FOR ALL USER INPUT (EXCEPT ABORTING
C   PRINTOUTS).  THE CALLING PARAMETERS ARE:
C
C   RETURN A1- IF THE USER TYPES X (ABORT EXIT).
C
C   RETURN A2- IF THE USER TYPES H, HE, HEL, OR HELP, PRECEDED
C               BY A CALL TO HELP WITH PARAMETER JHELP (DESCRIBED
C               BELOW).
C
C   JFLAG - INPUT TYPE DEFINED AS:
C             1 - NON-NEGATIVE INTEGER
C             2 - REAL NUMBER
C             3 - Y OR N (RETURNS 1 OR 0)
C             4 - CHARACTER STRING FROM TABLE
C             5 - CHARACTER STRING (MAX 10) OF USER'S CHOICE
C
C   TABLE - FOR JFLAG=4, AN ARRAY OF LEGAL INPUT OPTIONS,
C             DIMENSIONED AT MAX.
C
C   TAB2 - FOR JFLAG=5, THE ARRAY INTO WHICH THE USER'S INPUT
C           STRING IS STORED.
C

```

```

C      MIN - FOR JFLAG=1 OR 2, THE LOWER LIMIT OF ACCEPTABLE
C      RANGE OF DATA.
C
C      MAX - FOR JFLAG=1 OR 2, THE UPPER LIMIT OF ACCEPTABLE
C      RANGE OF DATA.
C      - FOR JFLAG=4, THE NUMBER OF INPUT OPTIONS (DIMENSION)
C      OF ARRAY TABLE.
C      - FOR JFLAG=5, THE MAXIMUM NUMBER OF CHARACTERS
C      ALLOWED IN THE INPUT STRING.
C
C      JHELP - THE REFERENCE NUMBER IN THE HELP FILE WHICH PROVIDES
C      ASSISTANCE FOR THIS PROMPT.
C
C      INT - FOR FLAG=1, THE INTEGER RESULT.
C      - FOR FLAG=2, THE REAL RESULT.
C      - FOR FLAG=3, 0 FOR N, 1 FOR Y.
C      - FOR FLAG=4, THE OPTION NUMBER SELECTED OR POSITION
C      IN TABLE (1 <= INT <= MAX).
C      - FOR FLAG=5, THE NUMBER OF CHARACTERS IN TAB2 WHICH
C      THE USER INPUT.
C
C      EQUIVALENCE (JMIN,AMIN),(JMAX,AMAX),(JINT,AINT)
C
C      CHARACTER*1 BL,COM,X,AHELP(5),CH(80),CHAR
C      CHARACTER*1 TAB2(10),DEC(10)
C      CHARACTER*10 TABLE(12)
C
C      INTEGER MASK,TITLE
C      COMMON /ALL/ J,KPR,KSD,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
C      COMMON /HINT/ JHINT
C
C      DATA BL,COM,X,AHELP/' ',' ','X','H','E','L','P',' '/
C
C      J IS NEXT POSITION (1 THRU 80) OF NEXT INPUT CHARACTER STORED IN
C      CH. IF ZERO, NO INPUT IS EXPECTED FROM TERMINAL. IF POSITIVE,
C      DATA IN CH IS REMAINDER FROM NEXT CALL.
C
C      IF (J.EQ. 0) READ (*,5) CH
C      5 FORMAT(80A1)
C
C      IF NEXT CHARACTER IS X FOLLOWED BY BLANK OR COMMA, USE ABORT
C      EXIT.
C
C      CHAR=CH(J+1)
C      IF (CHAR.NE. X) GOTO 10
C      CHAR=CH(J+2)
C      IF (CHAR.EQ. BL) GOTO 8
C      IF (CHAR.NE. COM) GOTO 10
C      J=J+2
C      RETURN 1
C      8 J=0

```

```

        RETURN 1
C
C IF USER TYPED HELP OR ANY PART OF IT, CALL HELP TO PROVIDE
C ASSISTANCE. THEN RETURN TO REPRINT THE PROMPT.
C
    10 IF (CHAR .NE. AHELP(1)) GOTO 15
        DO 13 K=2,5
        CHAR=CH(J+K)
        IF (CHAR .EQ. BL) GOTO 14
        IF (CHAR .NE. AHELP(K)) GOTO 15
    13 CONTINUE
    14 CALL HELP(JHELP)
        J=0
        RETURN 2
C
C
    15 GOTO (20,100,500,700,800),JFLAG
C
C READ FIRST CHARACTER OF INTEGER. IGNORE BLANK OR ONE PLUS.
C ERROR ON - OR . OR , OR ANY LETTER.
C
    20 KP=0
    30 J=J+1
        IF (J .GT. 80) GOTO 1000
        CALL FIND(CH(J),KODE)
        IF (KODE .LT. 1 .OR. KODE .GT. 14) GOTO 1000
        IF (KODE .LT. 11) GOTO 40
        KODE=KODE-10
        GOTO (30,1000,1000,35),KODE
C
C ALLOW ONLY ONE PLUS SIGN
C
    35 IF (KP .EQ. 1) GOTO 1000
        KP=1
        GOTO 30
C
C INITIALIZE NUMBER. THEN GET NEXT DIGIT. BLANK OR COMMA IS DELIMITER.
C
    40 INT=KODE-1
    50 J=J+1
        IF (J .GT. 80) GOTO 1000
        CALL FIND(CH(J),KODE)
        IF (KODE .LT. 1 .OR. KODE .GT. 13) GOTO 1000
        IF (KODE .LT. 11) GOTO 60
        GOTO 70
C
C CONVERT DIGIT. GET NEXT NUMBER
C
    60 INT=INT*10 + KODE-1
        GOTO 50
C
C CHECK RANGE (UNLESS NOT REQUIRED).

```

```

C
70 IF (MIN .EQ. 0 .AND. MAX .EQ. 0) GOTO 80
   IF (INT .LT. MIN .OR. INT .GT. MAX) GOTO 1000
C
C ASSURE PROPER TERMINATION.
C IF COMMA, MORE DATA FOLLOWS FOR NEXT PROMPT. IF BLANK, KEEP
C LOOKING. ANYTHING BUT BLANK OR A COMMA IS ILLEGAL.
C
80 IF (CH(J) .EQ. COM) RETURN
82 J=J+1
85 IF (J .GT. 80) GOTO 90
86 IF (CH(J) .EQ. COM) RETURN
   IF (CH(J) .NE. BL) GOTO 1000
   GOTO 82
90 J=0
   RETURN
C
C READ FIRST REAL CHARACTER. , IS ERROR. SET FLAG FOR OTHERS.
C KD, KM, AND KP ARE SET TO 1 UPON OCCURENCE OF DECIMAL, MINUS
C SIGN, OR PLUS SIGN.
C
100 KD=0
    KM=0
    KP=0
    FACT=.1
C
110 J=J+1
    IF (J .GT. 80) GOTO 1000
    CALL FIND(CH(J),KODE)
    IF (KODE .LT. 1 .OR. KODE .GT. 15) GOTO 1000
    IF (KODE .LT. 11) GOTO 160
    KODE=KODE-10
    GOTO (150,1000,120,140,130),KODE
C
C SET UP FOR FRACTION
C
120 IF (KD .EQ. 1) GOTO 1000
    KD=1
    GOTO 110
C
C SET UP NEGATIVE NUMBER
C
130 IF (KD+KP+KM .NE. 0) GOTO 1000
    KM=1
    GOTO 110
C
C PLUS IS NOT ALLOWED AFTER ANYTHING ELSE
C
140 IF (KP+KD+KM .NE. 0) GOTO 1000
    KP=1
    GOTO 110
C

```

```

C NO BLANKS AFTER A DECIMAL
C
150 IF (KD) 110,110,1000
C
C SET UP FIRST DIGIT
C
160 AINT=KODE-1
    IF (KD .EQ. 0) GOTO 170
    AINT=AIN*FACT
    FACT=FACT*.1
C
C SET NEXT CHARACTER
C
170 J=J+1
    IF (J .GT. 80) GOTO 1000
    CALL FIND(CH(J),KODE)
    IF (KODE .LT. 1. OR. KODE .GT. 13) GOTO 1000
    IF (KODE .LT. 11) GOTO 190
    IF (KODE .EQ. 13) GOTO 180
    GOTO 300
C
C FIX DECIMAL POINT
C
180 IF (KD .EQ. 1) GOTO 1000
    KD=1
    GOTO 170
C
C INSERT NEXT NUMBER
C
190 IF (KD .EQ. 1) GOTO 200
    AINT=AIN*10.0 + KODE-1
    GOTO 170
200 AINT=AIN + (KODE-1)*FACT
    FACT=FACT*.1
    GOTO 170
C
C SET NEGATIVE IF A MINUS SIGN WAS ENCOUNTERED. INT IS THEN
C SET AS THE RESULT BY SETTING TO JINT WHICH IS EQUIVALENCED
C TO AINT. ALL THIS IS NECESSARY TO AVOID CONVERSION WHEN
C STORING THE RESULT IN INT. SIMILARLY, NO CONVERSION IS
C WANTED WHEN LOOKING AT MIN AND MAX.
C
300 IF (KM .EQ. 1) AINT=-AINT
    INT=JINT
    JMAX=MAX
    JMIN=MIN
    IF (AMAX .EQ. 0.0. AND. AMIN .EQ. 0.0) GOTO 80
    IF (AINT .LT. AMIN .OR. AINT .GT. AMAX) GOTO 1000
    GOTO 80
C
C CHECK FOR Y OR N
C

```

```

500 J=J+1
    IF (J .GT. 80) GOTO 1000
    IF (CH(J) .EQ. BL) GOTO 500
    CALL FIND(CH(J),KODE)
    J=J+1
    IF (KODE-16) 1000,510,520
510 INT=1
    GOTO 86
520 INT=0
    GOTO 86
C
C CHECK FOR KEYWORD FROM 'TABLE'
C
700 DO 750 INT=1,MAX
    TEMP=TABLE(INT)
    READ(TABLE(INT),710) DEC
710 FORMAT(10A1)
    K=J
    DO 730 L=1,10
    K=K+1
    IF (DEC(L) .EQ. CH(K)) GOTO 730
C
C A MISS. IF FIRST CHARACTER, TRY NEXT IN LIST
C
    IF (L .EQ. 1) GOTO 750
C
C OK TO ABBREVIATE IF NEXT IS BLANK (OR COMMA)
C
    IF (CH(K) .NE. BL) GOTO 720
    J=K
    GOTO 82
720 IF (CH(K) .NE. COM) GOTO 750
    J=K
    GOTO 80
730 CONTINUE
    J=K
    GOTO 82
750 CONTINUE
    GOTO 1000
C
C READ LITERAL STRING
C
800 K=0
    INT=0
810 K=K+1
    J=J+1
    IF (CH(J) .NE. BL) GOTO 815
812 IF (K .GT. MAX) GOTO 820
    TAB2(K)=CH(J)
    K=K+1
    J=J+1
    IF (CH(J) .EQ. BL) GOTO 812

```

```

C
C
200 K=-10
    IF(K .LE. 2) GOTO 201
    K=K-1
    IF(K .GT. 6) K=K-3
201 INQUIRE(FILE=FILES(K),OPENED=OPENED)
    IF(OPENED) CLOSE (-10)
    IF(K .EQ. 3 .OR.
*   K .EQ. 6 .OR.
*   K .EQ. 10 .OR.
*   K .EQ. 11) GOTO 202
    OPEN(-10, FILE = FILES(K), STATUS='NEW')
    GOTO 100
202 OPEN(-10, FILE = FILES(K), STATUS='NEW', FORM='BINARY')
    GOTO 100
    END
    BLOCKDATA
    CHARACTER*7 AFID,SEID
    CHARACTER*7 SEQID,LEQID
    COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)
C
    COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
    COMMON /COSTIO/ IOIN,IOUT,NTH
C
    COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
    COMMON /ERR/ JERR,KOUNT
C
    CHARACTER*8 FIELDS
    COMMON /EX/ FIELDS(8)
C
    COMMON /OVER/ JABT,IO2,IO4A,JN,NMAX,LAST
C
    INTEGER MASK,TITLE
    COMMON /ALL/ NOTHER,KPR,KSQ,KLI,NHASK,NTITL,MASK(10),TITLE(10)
C
    COMMON /HINT/ JHINT
C
    CHARACTER*1 COLS(80)
    COMMON /MODIF/ IT,COLS
C
    CHARACTER*1 XMASK,NEWT
    COMMON /JJF/ XMASK(10),NEWT(10)
C
    COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
    CHARACTER*7 SFSE,SFAFSC
    COMMON /SHAREX/ TSFL(7,40),PSM(7,40),SFSE(2,7,40),SFAFSC(5,7,40),
*           NSAFSC(7,40),NSFSE(7,40),FHBMA(40),HFAC(40)

```

```

2 WRITE(*,5)
5 FORMAT(/,' R+M FILE NOT ATTACHED. ')
      BOTO 500

C
C PROMPT FOR TIME, PROB, OR MFHBMA. THEN UNLESS MFHBMA, PROMPT
C FOR TASK.
C
C JT = 1-5 FOR SHOP
C      6-12 FOR FLIGHTLINE
C      13 FOR MFHBMA IF IT=3
C      13 FOR CND IF IT = 1 OR 2
C
10 IF (NOTHER .EQ. 0) WRITE(*,20)
20 FORMAT(/,' R+M VARIABLE? ')
      CALL INP(4,VARY,0,0,27,80,IV,*500,*10)
      IF (IV-26) 25,30,35
25 IT = 3
      IF (IV .LE. 24) IT = 2
      IF (IV .LE. 12) IT = 1
      JT = 13
      IF (IV .LE. 23) JT = IV - 10
      IF (IV .LE. 17) JT = IV - 11
      IF (IV .LE. 12) JT = IV
      IF (IV .EQ. 4 .OR. IV .EQ. 16) JT = 7
      IF (IV .EQ. 6 .OR. IV .EQ. 18) JT = 4
      IF (IV .EQ. 16) JT = 5
      PROB = IT.EQ.2
      SHOP = JT.LE.5
      TEST = JT.EQ.4 .OR. JT.EQ.5
      BOTO 46
30 CALL DEFINE
      BOTO 10
35 CALL SET
      BOTO 10

C
46 IF (NTITL .EQ. -1) BOTO 47
      NT=NTITL
      BOTO 49
47 IF (NOTHER .EQ. 0) WRITE(*,48)
48 FORMAT(/,' NEW TITLE? ')
      CALL INP(5,0,NEWT,0,10,130,NT,*10,*47)

C
C SPECIAL CASE IF JT=13 AND IT=2 (FLIGHTLINE + SHOP CND PROBABILITY
C COMBINED).
C
49 IF (JT .NE. 13 .OR. IT .EQ. 3) BOTO 50
      CALL MODCND(NEWT)
      BOTO 500

C
C FOR PROBABILITY MODIFICATION, FACTOR IS NECESSARY (KP=2),
C OTHERWISE BIAS (1) OR REPLACE (3) IS ALLOWED.
C

```

```

50 IF (.NOT. PROB) GOTO 51
   KP=2
   GOTO 70
51 IF (NOTHER .EQ. 0) WRITE(*,60)
60 FORMAT(/, ' TYPE?      ')
   CALL INP(4,PERT,0,0,3,100,KP,*10,*50)
70 IF (NOTHER .EQ. 0) WRITE(*,80) PERT(KP)
80 FORMAT(/,1X,A7,'=      ')
   CALL INP(2,0,0,0,0,110,VAL,*10,*70)
C
   IF (NMASK .EQ. -1) GOTO 90
   MK=NMASK
   GOTO 105
90 IF(NOTHER .EQ. 0) WRITE(*,100)
100 FORMAT(/, ' MASK=      ')
   CALL INP(5,0,XMASK,0,7,120,MK,*10,*90)
C
105 IF (KLI .EQ. -1) GOTO 110
   IL=KLI
   GOTO 121
110 IF (NOTHER .EQ. 0) WRITE(*,120)
120 FORMAT(/, ' DO YOU WANT A LISTING OF THE CHANGED ITEMS?      ')
   CALL INP(3,0,0,0,0,2,IL,*10,*110)
C
C THE SCHEME IS TO READ JIN ONE RECORD AT A TIME AND MODIFY THE
C DATA AND WRITE TO JOUT.
C   KOUNT = PRINT LINE COUNT
C   KTOT = TOTAL CHANGES OF SELECTED TASK
C   LTOT = TOTAL CHANGES OF AFFECTED LRUS
C   NSUB = BUMPED FOR EACH SUBSYSTEM IN THE MASK
C   ITRUNC = NUMBER OF MODIFICATIONS TRUNCATED TO LESS THAN
C           DESIRED FACTOR
C   INONE = NUMBER OF MODIFICATIONS WHICH COULD NOT BE CHANGED
C
121 REWIND 11
   KOUNT=0
   KTOT=0
   LTOT=0
   NSUB=0
   ITRUNC=0
   INONE=0
C
C READ FIRST RECORD.  INSERT TITLE.  SKIP TO WRITE OUTPUT.
C
   READ(11,140) COLS
   DO 125 K=1,10
     COLS(70+K)=NEW(K)
125 CONTINUE
   GOTO 145
C
130 READ(11,140,END=480) COLS
140 FORMAT(80A1)

```

```

C
C KEEP READING AND WRITING TILL WE GET PROPER CARD FORMAT.
C
C     IF (COLS(1) .NE. C1(IT)) GOTO 145
C
C COLUMN IS TRUE IF THE CARD IS OF THE SHOP/FL THAT MATCHES WHAT
C WE'RE MODIFYING.
C
C     COLUM=COLS(2).EQ.C2(JT)
C
C MODIFYING FLIGHTLINE PROB WILL AFFECT SHOP AS WELL.
C
C     IF (PROB .AND. .NOT. SHOP) GOTO 142
C
C OTHERWISE CARD TYPE MUST BE EXACT.
C
C     IF (.NOT. COLUM) GOTO 145
142 CALL MATCH(MK,XMASK,6,COLS(4),*145)
C     GOTO 150
145 WRITE(12,140) COLS
C     GOTO 130
C
C FIND THE SUBSYSTEM FOR THIS LRU.
C
150 IF (COLUM) GOTO 155
C     WRITE(TSUB,360) (COLS(K),K=4,8)
C     DO 152 J=1,NSUB
C     IF (SUB(J) .EQ. TSUB) GOTO 154
152 CONTINUE
C     WRITE(*,153) NSUB
153 FORMAT(/,' LCCIM SYSTEM ERROR 1',15)
C
C GET FACTOR TO PASS DOWN FROM SUBSYSTEM TO SHOP. ADD TO
C LRU'S AFFECTED.
C
154 F=FACT(J)
C     DO 156 J=1,5
C     CALL SLAP(J,STRIP(J)*F)
156 CONTINUE
C     LTOT=LTOT+1
C     GOTO 145
C
C PRINT HEADER FOR LISTING OF EQUIPMENTS MODIFIED AS REQUESTED.
C
155 KTOT=KTOT+1
C     IF (KTOT .EQ. IL) WRITE(*,157)
157 FORMAT(/,' EQUIP',5X'RMBASE',5X'RMPT')
C
C     OLD=STRIP(JT)
C
C     IF (KP-2) 170,180,190
C

```

```

C BIAS
C
  170 GNU=OLD+VAL
      BOTO 200
C
C FACTOR
C
  180 GNU=OLD+VAL
      BOTO 200
C
C REPLACE
C
  190 GNU=VAL
C
  200 IF (.NOT. PROB) BOTO 470
      IF (SHOP .AND. .NOT. TEST) BOTO 220
C
C FOR FLIGHTLINE OR TEST STATION PROBABILITIES, REPLACE FACTOR
C (VAL) IF RESULT > .95. COUNT TRUNCATIONS.
C
      IF (GNU .LT. 1.0) BOTO 210
      GNU=.95
      VAL=GNU/OLD
      ITRUNC=ITRUNC+1
  210 IF (TEST) BOTO 470
      BOTO 300
C
C FOR SHOP PROBABILITIES, SET JT1 AND JT2 TO THE OTHER TWO SHOP
C TASKS AND EXTRACT THE PROBABILITIES.
C
  220 JT1=MOD(JT,3)+1
      JT2=MOD(JT+1,3)+1
      S1=STRIP(JT1)
      S2=STRIP(JT2)
      F=S1+S2
C
C IF VALUE FOR CHANGING IS THE ONLY ONE, IT CANNOT BE CHANGED.
C
      IF (F .GT. 0.0) BOTO 260
      INONE=INONE+1
      BOTO 475
C
C IF THE REQUIRED CHANGE WOULD PUT THE TOTAL OVER 1.0 TRUNCATE.
C
  260 IF (GNU .LT. F+OLD) BOTO 270
      ITRUNC=ITRUNC+1
      GNU=.95*(F+OLD)
  270 F=(F+OLD-GNU)/F
      S1=S1*F
      S2=S2*F
C

```

```

C  REPLACE THE OTHER TWO.
C
280  CALL SLAP(JT1,81)
      CALL SLAP(JT2,82)
      GOTO 470
C
C  MOD TO FLIGHTLINE PROB
C
300  IF (JT-8) 310,320,305
C
C  USER MODIFY OTHER THAN TROUBLESHOOT OR CANNOT DUPLICATE.
C  BMAX IS TROUBLESHOOT PROBABILITY.
C  IF MODIFYING R+R OR MAC WOULD MAKE IT EXCEED THE MAX, WE
C  TRUNCATE IT. THEN MODIFY THE OTHER THREE.
C
305  BMAX=STRIP(7)
      IF (BNU .LT. BMAX) GOTO 308
      VAL=.95*BMAX/OLD
      BNU=OLD*VAL
      ITRUNC=ITRUNC+1
308  IF (JT .EQ. 9 .OR. JT .EQ. 11) GOTO 330
      GOTO 340
C
C  USER MODIFY TROUBLESHOOT. CANNOT DUPLICATE BECOMES COMPLEMENT.
C  R+R AND MAC MODIFIED BY SAME AMOUNT AS TROUBLESHOOT.
C
310  CALL SLAP(8,1.0-BNU)
      F=VAL
312  DO 315 J=9,12
      CALL SLAP(J,STRIP(J)*F)
315  CONTINUE
      GOTO 350
C
C  USER MODIFY CND. TROUBLESHOOT BECOMES COMPLEMENT.
C  F IS FACTOR TO PASS DOWN TO SHOP, REPRESENTING AMOUNT OF R+R
C  CHANGE.
C
320  TS=1.0-BNU
      TSOLD=STRIP(7)
      CALL SLAP(7,TS)
      F=TS/TSOLD
      GOTO 312
C
C  HERE WE MODIFY THE THREE TASKS OUT OF THE LAST FOUR (9,10,11,12)
C  COMPOSED OF R+R, MAC, VR+R, VMAC, EXCLUDING THE ONE WE SELECTED
C  TO MODIFY. F IS FACTOR TO PASS DOWN TO SHOP.
C
330  CALL SLAP(20-JT,BNU)
      CALL SLAP(10,BMAX-BNU)
      CALL SLAP(12,BMAX-BNU)
      F=VAL
      GOTO 350

```

```

C
340 CALL SLAP(22-JT,GNU)
    RR=6MAX-GNU
    RROLD=STRIP(9)
    CALL SLAP(9,RR)
    CALL SLAP(11,RR)
    F=RR/RROLD
C
C SET FACTOR FOR LRUS
C
350 NSUB=NSUB+1
    WRITE(TSUB,360) (COLS(K),K=4,8)
360 FORMAT(5A1)
    FACT(NSUB)=F
    SUB(NSUB)=TSUB
C
C FOR ALL MODIFICATIONS, THE SELECTED FIELD IS REPLACED HERE.
C
470 CALL SLAP(JT,GNU)
475 IF (IL .EQ. 0) GOTO 145
    CALL ABORT(KOUNT,*145)
    PFORM(2)=FORM(IT)
    PFORM(4)=FORM(IT)
    WRITE(*,PFORMX) (COLS(K),K=4,10),OLD,GNU
    GOTO 145
C
C
480 WRITE(*,490) KTOT
490 FORMAT(/,1X,I4,' CHANGES.')
    IF (ITRUNC .NE. 0) WRITE(*,495) ITRUNC
495 FORMAT(1X,I4,' TRUNCATIONS.')
    IF (INONE .GT. 0) WRITE(*,496) INONE
496 FORMAT(1X,I4,' NOT CHANGED.')
    IF (LTOT .GT. 0) WRITE(*,497) LTOT
497 FORMAT(1X,I4,' LRUS CHANGED.')
C
500 RETURN
    END
    FUNCTION STRIP(JT)
C
C THIS FUNCTION EXTRACTS A VALUE FROM AN R+M CARD PASSED IN
C ARRAY COLS.
C CALLED BY MODIFY, MODCND
C
C JT - SPECIFIES THE COLUMNS FROM WHICH THE DATA IS TO BE
C EXTRACTED BY USING STARTING COLUMN LBEG.
C
C IT - SELECTS THE PROPER FORMAT.
C 1 - TIME
C 2 - PROBABILITY
C 3 - MFHBMA
C

```

```

      DIMENSION LBEG(13)
      CHARACTER*10 FORM(3),TEMP
C
      CHARACTER*1 COLS
      COMMON /MODIF/ IT,COLS(80)
C
      DATA LBEG/19,25,31,49,55,13,19,25,31,37,43,49,14/
      DATA FORM/'(F6.1)', '(F6.4)', '(F6.1)'/
C
C  GET STARTING AND ENDING COLUMNS.  COMBINE, THEN DECODE.
C
      J1=LBEG(JT)
      J2=J1+5
      WRITE(TEMP,10) (COLS(K),K=J1,J2)
10  FORMAT(6A1)
      READ(TEMP,FORM(IT)) STRIP
      RETURN
      END
      SUBROUTINE SLAP(JT,C)
C
C  THIS SUBROUTINE PERFORMS THE REVERSE FUNCTION OF STRIP,
C  PLACING A VALUE INPUT ONTO AN R+M CARD.
C  CALLED BY MODIFY, MODCND
C
C  JT - SPECIFIES THE COLUMNS ONTO WHICH THE DATA IS TO BE CODED
C  BY USING STARTING COLUMN LBEG.
C
C  C - VALUE TO BE STORED.
C
C  IT - SELECTS THE PROPER FORMAT.
C       1 - TIME
C       2 - PROBABILITY
C       3 - MFHBMA
C
      DIMENSION LBEG(13)
      CHARACTER*10 FORM(3),TEMP
C
      CHARACTER*1 COLS
      COMMON /MODIF/ IT,COLS(80)
C
      DATA LBEG/19,25,31,49,55,13,19,25,31,37,43,49,14/
      DATA FORM/'(F6.1)', '(F6.4)', '(F6.1)'/
C
C  GET STARTING AND ENDING COLUMNS.  ENCODE, THEN PUT ON CARD.
C
      J1=LBEG(JT)
      J2=J1+5
      WRITE(TEMP,FORM(IT)) C
      READ(TEMP,10) (COLS(K),K=J1,J2)
10  FORMAT(6A1)

```

```

      RETURN
      END
      SUBROUTINE MODCND (NENT)
C
C   THIS ROUTINE IS A SPECIAL CASE OF MODIFY WHERE THE USER HAS
C   SPECIFIED 'CND', MEANING MODIFY BOTH SHOP AND FLIGHTLINE
C   PROBABILITY, WITH RESULTANT MODIFICATION TO MFHBMA.
C   CALLED BY MODIFY
C
C   NENT = TITLE OF NEW FILE
C   FCND = FLIGHTLINE CANNOT DUPLICATE FACTOR
C   CFCND = COMPLEMENT OF FCND
C   SCND = SHOP CANNOT DUPLICATE FACTOR
C   CSCND = COMPLEMENT OF SCND
C   PK = SUM OF SHOP CANNOT DUPLICATE FOR SUBSYSTEM
C           -OR-
C   OLD FLIGHTLINE CND PROBABILITY
C   JSUB = NUMBER OF SUBSYSTEM CHANGES
C   JLRU = NUMBER OF SHOP CHANGES
C   KT = CARD TYPE COUNTER
C   JHIT = 1 IF CARD READ IS OF INTEREST
C
      DIMENSION PK(32)
      CHARACTER*6 FIELD
      CHARACTER*1 SCOL(5),NENT(10)
      CHARACTER*10 TSUB,SUB(32)
      CHARACTER*2 SF(3),CC
C
      CHARACTER*1 COLS
      COMMON /MODIF/ IT,COLS(80)
C
      INTEGER MASK,TITLE
      COMMON /ALL/ NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
      CHARACTER*1 XMASK,XTITLE
      COMMON /JJF/ XMASK(10),XTITLE(10)
C
      EQUIVALENCE (COLS(4),SCOL(1))
C
      DATA SF/'PF','PS','MF'/
C
      IT=2
      10 IF (NOTHER .EQ. 0) WRITE(*,20)
      20 FORMAT(/,' FLIGHTLINE CND FACTOR = ')
      CALL INP(2,0,0,0.0,99999.0,110,FCND,*230,*10)
      CFCND=1.0-FCND
      30 IF (NOTHER .EQ. 0) WRITE(*,40)
      40 FORMAT(/,' SHOP CND FACTOR = ')
      CALL INP(2,0,0,0.0,99999.0,110,SCND,*230,*30)
      CSCND=1.0-SCND
      IF (NMASK .EQ. -1) GOTO 50
      MK=NMASK

```

```

      GOTO 65
50 IF (NOTHER .EQ. 0) WRITE(*,60)
60 FORMAT(/,' MASK=      ')
      CALL INP(5,0,XMASK,0,7,120,MK,*230,*50)
C
C FIRST GET OLD SCND TOTALS PER SUBSYSTEM
C
      65 REWIND 11
         NSUB=0
      70 READ(11,80,END=105) CC,SCOL,FIELD
      80 FORMAT(A2,1X,5A1,16X,A6)
         IF (CC .NE. SF(2)) IF (NSUB) 70,70,110
         CALL MATCH(MK,XMASK,5,COLS(4),*70)
         WRITE(TSUB,170) SCOL
         READ(FIELD,85) VAL
      85 FORMAT(F6.4)
C
C IF SUBSYSTEM IS ALREADY TALLIED, ADD TO IT.
C
         IF (NSUB .EQ. 0) GOTO 100
         DO 90 J=1,NSUB
         IF (TSUB .NE. SUB(J)) GOTO 90
         PK(J)=PK(J)+VAL
         GOTO 70
      90 CONTINUE
     100 NSUB=NSUB+1
         SUB(NSUB)=TSUB
         PK(NSUB)=VAL
         GOTO 70
C
     105 WRITE(*,106)
     106 FORMAT(/,' NO DATA')
         NOTHER=0
         GOTO 50
C
C NOW COPY JIN TO JOUT, CHANGING PF, PS, AND MF CARDS
C
     110 REWIND 11
         REWIND 12
         JSUB=0
         JLRU=0
         JFHB=0
         KT=1
         JHIT=0
C
C FIRST READ HEADER CARD, INSERT TITLE, SKIP TO WRITE OUTPUT.
C
         READ(11,120) CC,(COLS(K),K=3,80)
     120 FORMAT(A2,78A1)
         DO 130 K=1,10
            COLS(70+K)=NEWT(K)
     130 CONTINUE

```

```

      GOTO 150
C
140 READ (11,120,END=225) CC,(COLS(K),K=3,80)
      IF (KT .GT. 3) GOTO 150
      IF (CC .EQ. SF(KT)) GOTO 160
C
C IF NOT AT CARD OF INTEREST, SIMPLY COPY IT. IF WE WERE,
C START LOOKING FOR NEXT CARD OF INTEREST.
C
      IF (JHIT .EQ. 0) GOTO 150
      JHIT=0
      KT=KT+1
      IF (KT .EQ. 3) IT=3
      IF (CC .EQ. SF(KT)) GOTO 160
150 WRITE(12,120) CC,(COLS(K),K=3,80)
      GOTO 140
C
C CHECK MASK
C
160 JHIT=1
      CALL MATCH(MK,XMASK,5,COLS(4),*150)
C
C FIND CORRESPONDING PK FOR THIS EQUIP
C
      WRITE(TSUB,170) SCOL
170 FORMAT(5A1)
      DO 180 J=1,NSUB
      IF (TSUB .EQ. SUB(J)) IF (KT-2) 200,210,220
180 CONTINUE
      WRITE(*,190) NSUB
190 FORMAT(/,' LCCIM SYSTEM ERROR 2',15)
C
C REPLACE FLIGHTLINE PROBABILITIES. PK CONTAINS SUM OF SHOP CND
C
200 PC=STRIP(8)
      FACT=1.0-CFCND*PC
      DIFF=CSCND*PK(J)
C
C RESET PK TO PREVIOUS CND FOR FLIGHTLINE
C
      PK(J)=PC
      CALL SLAP(8,(FCND*PC+DIFF)/FACT)
      CALL SLAP(7,(STRIP(7)-DIFF)/FACT)
      TEMP=(STRIP(9)-DIFF)/FACT
      CALL SLAP(9,TEMP)
      CALL SLAP(11,TEMP)
      TEMP=STRIP(10)/FACT
      CALL SLAP(10,TEMP)
      CALL SLAP(12,TEMP)
      JSUB=JSUB+1
      GOTO 150
C

```

C REPLACE SHOP PROBABILITIES. PK IS OLD FLIGHTLINE CND

C

```
210 FACT=1.0-CFCND*PK(J)
    CALL SLAP(1,STRIP(1)/FACT)
    CALL SLAP(2,STRIP(2)*SCND/FACT)
    CALL SLAP(3,STRIP(3)/FACT)
    CALL SLAP(4,STRIP(4)/FACT)
    CALL SLAP(5,STRIP(5)/FACT)
    JLRU=JLRU+1
    BOTO 150
```

C

C REPLACE FHBMA. PK IS STILL OLD FLIGHTLINE CND

C

```
220 CALL SLAP(13,STRIP(13)/(1.0-CFCND*PK(J)))
    JFHB=JFHB+1
    BOTO 150
```

C

```
225 WRITE(*,226) JSUB,JLRU
226 FORMAT(/,I4,I4,' FLIGHTLINE CHANGES'/I4,' SHOP CHANGES')
    IF (JSUB.EQ. NSUB .AND. NSUB.EQ. JFHB) BOTO 230
    WRITE(*,227) NSUB,JFHB
227 FORMAT(' LCCIM SYSTEM ERROR 3',2I5)
    NOTHER=0
230 RETURN
    END
    SUBROUTINE RMODEL
```

C

C THIS SUBROUTINE INITIATES R+M MODEL CALCULATIONS. DATA IS  
C READ FROM CARDS TO /TEMPRM/ AND /SHARE/, THEN CONVERTED TO  
C OUTPUTS AND STORED IN /RAM/. IF THE USER WANTED TO COMPARE  
C WITH A PERTURBED FILE, THE PROCESS IS REPEATED BUT STORED IN  
C SUBSCRIPT 2 OF /RAM/.  
C CALLED BY MAIN

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

```
    DIMENSION TRIXS(50),TRIXL(50)
```

```
    COMMON /OVER/ JABT,I02,I04A,JN,NMAX,LAST
```

```
    COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
```

```
    CHARACTER*7 SFSE,SFAFSC
```

```
    COMMON /SHARE/ TSFL(7,40),PSM(7,40),SFSE(2,7,40),SFAFSC(5,7,40),
    *      NSAFSC(7,40),NSFSE(7,40),FHBMA(40),HFAC(40)
```

```
    CHARACTER*7 LSAFSC,LSE
```

```
    COMMON /TEMPRM/ TLSHOP(5,120),PLRR(5,120),LSAFSC(5,5,120),
    *      LSE(2,5,120),NLAFSC(5,120),NLSE(5,120),
    *      WEIGHT(120),NSRU(120)
```

```
    COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
```

```

      CHARACTER*7 AFID,SEID
      CHARACTER*7 SEQID,LEQID
      COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)

C
C
C
C   LOOP IS INITIALLY SET TO 1.  LAST(1) IS ZERO THE FIRST TIME
C   BUT IS SET TO THE R+M FILE UNIT NUMBER USED FOR INPUT IN
C   RMREAD.  IF LATER THE USER USES THE SAME FILE, THE MODEL
C   WILL NOT RECALCULATE.  (THE EXCEPTION IS WHEN MODCST MODIFIES
C   /RAM/, AND LAST IS SET TO ZERO, OR MODIFY CREATES A NEW R+M
C   FILE.
C
      LOOP=1
      IF (LAST .EQ. 1) GOTO 100
      MMHIO=4
      CALL CHECK(-4,*998)
      CALL CHECK(-7,*998)
      LAST = 1
      NSE=0
      NAF=0
      10 CALL RMREAD(LOOP+10,*200)

C
C FIRST GET SUBSYSTEM FLIGHTLINE DATA (POSITIONS 2, 5, ADM 8 IN SDAT)
C
      DO 30 JSUB=1,NSUB
      FH=FHBMA(JSUB)
      SDAT(JSUB,8,LOOP)=FH
      PKFH=1000./FH
      HF = HFAC(JSUB)
      TTR=0.0
      EMM=0.0

C
C INITIALIZE MMH MATRIX
C
      DO 12 J=1,50
      TRIXS(J)=0.0
      12 CONTINUE

C
C ADD UP MTTR AND MMH/KFH FOR ALL 7 FLIGHTLINE TASKS.
C
      DO 20 J=1,7
      X = TSFL(J,JSUB) * PSM(J,JSUB)
      TTR=TTR+X
      N=NSAFSC(J,JSUB)
      IF (N .EQ. 0) GOTO 20
      EMM=EMM+X*N

C
C FOR EACH AFSC REQUIRED FOR THE TASK, COLLECT STATS IN 'FILE'.
C
      DO 15 K=1,N
      CALL FILE(SFAFSC(K,J,JSUB),X/FH,1,LOOP,JAF,*200)
      TRIXS(JAF)=TRIXS(JAF)+X/FH

```

```

15 CONTINUE
20 CONTINUE
C
  SDAT(JSUB,2,LOOP) = YTR
  SDAT(JSUB,5,LOOP) = ENH * PKFH
  TTRS=0.0
  ENMS=0.0
C
C NOW LRU DATA (FOR THIS SUBSYSTEM)
C
  JLRU=KLRU(JSUB)
  NR=NUML(JSUB)
  NTOT=0
  DO 40 K=1,NR
    NTOT=NTOT+NSRU(JLRU)
    TTRL=0.0
    ENHL=0.0
C
C INITIALIZE MMH ARRAY
C
  DO 22 J=1,50
    TRIXL(J)=0.0
  22 CONTINUE
C
C ADD UP MTTR AND MMH/KFH, BUT ONLY FOR M, K, N TASKS.
C
  DO 30 M=1,5
    X = TLSHOP(M,JLRU) * PLRR(M,JLRU) * HF
    IF (M .LE. 3) TTRL = TTRL + X
    N=NLAFSC(M,JLRU)
    IF (N .EQ. 0) GOTO 30
    IF (M .LE. 3) ENHL=ENHL+X*N
C
C FOR EACH AFSC REQUIRED FOR THE TASK, COLLECT STATS IN 'FILE'.
C
  DO 25 J=1,N
    CALL FILE(LSAFSC(J,M,JLRU),X/FH,2,LOOP,JAF,*200)
    TRIXL(JAF)=TRIXL(JAF)+X/FH
  25 CONTINUE
  30 CONTINUE
C
  N=NLSE(4,JLRU)
  IF (N .EQ. 0) GOTO 37
  Y=(PLRR(4,JLRU)*TLSHOP(4,JLRU)+PLRR(5,JLRU)*TLSHOP(5,JLRU))/FH
C
C FOR EACH SE REQUIRED FOR THE TASK, COLLECT STATS IN FILE2.
C
  DO 35 J=1,N
    CALL FILE2(LSE(J,4,JLRU),TTRL/FH,Y,LOOP,*200)
  35 CONTINUE
C
C ASSIGN RAM DATA.

```

```

C
37 UDAT(JLRU,1,LOOP) = TTRL
   UDAT(JLRU,2,LOOP) = EMML + PKFH
   UDAT(JLRU,3,LOOP) = WEIGHT(JLRU)
   X = PLRR(1,JLRU)
   Y = PLRR(3,JLRU)
   UDAT(JLRU,4,LOOP) = X
   UDAT(JLRU,5,LOOP) = Y
   UDAT(JLRU,6,LOOP) = X + Y + PLRR(2,JLRU)
   UDAT(JLRU,7,LOOP) = NSRU(JLRU)
   TTRS = TTRS + TTRL
   EMMS = EMMS + EMML
C
C DUMP MMH MAXTRIX TO FILE MMHIO
C
   DO 38 J=1,NAF
   X=TRIXL(J)
   IF (X .GT. 0.0) WRITE(MMHIO) -JLRU,J,X
38 CONTINUE
   JLRU=JLRU+1
40 CONTINUE
C
C NOW STORE SUBSYSTEM SHOP AND TOTAL (1 AND 3 FOR MTTR, 4 AND 6 FOR MMH)
C NTOT IS NUMBER OF SRUS.
C
   SDAT(JSUB,1,LOOP) = TTRS
   SDAT(JSUB,4,LOOP) = EMMS + PKFH
   SDAT(JSUB,3,LOOP) = TTR + TTRS
   SDAT(JSUB,6,LOOP) = EMH + EMMS
   SDAT(JSUB,9,LOOP) = NTOT
C
C DUMP MMH MATRIX TO FILE MMHIO
C
   DO 45 J=1,NAF
   X=TRIXS(J)
   IF (X .GT. 0.0) WRITE(MMHIO) JSUB,J,X
45 CONTINUE
C
C FINALLY AVAILABILITY
C
   SDAT(JSUB,7,LOOP) = 1.0 / (1.0+TTR/FH)
50 CONTINUE
   DO 95 J=1,NAF
   ADAT(J,3,LOOP) = ADAT(J,1,LOOP) + ADAT(J,2,LOOP)
95 CONTINUE
C
C STORE PERTURBED DATA SIMILARLY, IF ANY.
C
100 IF (LOOP .EQ. 2 .OR. I02 .EQ. 0) RETURN
    LOOP=2
    MMHIO=7
    REWIND 7

```

```

C      DO 110 J=1,NAF
      DO 110 K=1,3
      ADAT(J,K,2)=0.0
110 CONTINUE
C
      DO 120 J=1,NSE
      DO 120 K=1,2
      EDAT(J,K,2)=0.0
120 CONTINUE
      GOTO 10
200 NOTHER=0
      JABT=1
      RETURN
998 WRITE(*,999)
999 FORMAT(///,' INTERNAL PROGRAM ERROR - RMODEL')
      STOP
      END
      SUBROUTINE FILE(AF,X,K,LOOP,J,*)
C
C THIS ROUTINE ADDS UP MMH/KFH FOR EACH AFSC.
C
      CHARACTER*7 AF
C
      CHARACTER*7 AFID,SEID
      CHARACTER*7 SEQID,LEQID
      COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)
C
      COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
      IF (NAF .EQ. 0) GOTO 20
      IF (NAF .LE. 50) GOTO 8
      WRITE(*,7)
      7 FORMAT(/,' NUMBER OF AFSCs EXCEEDS 50.')
      NOTHER=0
      RETURN 1
      8 DO 10 J=1,NAF
      IF (AF .EQ. AFID(J)) GOTO 30
10 CONTINUE
20 NAF=NAF+1
      ADAT(NAF,1,LOOP)=0.0
      ADAT(NAF,2,LOOP)=0.0
      AFID(NAF)=AF
      J=NAF
30 ADAT(J,K,LOOP) = ADAT(J,K,LOOP) + X
      RETURN
      END
      SUBROUTINE FILE2(SE,X,Y,LOOP,*)
C
C THIS ROUTINE ADDS UP MMH/FH FOR EACH SE.

```

```

C      CHARACTER*7 SE
C
C      CHARACTER*7 AFID,SEID
C      CHARACTER*7 SEQID,LEQID
C      COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)
C
C      COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
C      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
C      IF (NSE .EQ. 0) GOTO 20
C      IF (NSE .LE. 50) GOTO 8
C      WRITE(*,7)
C      7 FORMAT(/,' NUMBER OF SUPPORT EQUIPMENT EXCEEDS 50. ')
C      NOTHER=0
C      RETURN 1
C      8 DO 10 J=1,NSE
C        IF (SE .EQ. SEQID(J)) GOTO 30
C      10 CONTINUE
C      20 NSE=NSE+1
C        EDAT(NSE,1,LOOP)=0.0
C        EDAT(NSE,2,LOOP)=0.0
C        SEQID(NSE)=SE
C        J=NSE
C      30 EDAT(J,1,LOOP)=EDAT(J,1,LOOP)+X
C        EDAT(J,2,LOOP)=EDAT(J,2,LOOP)+Y
C      RETURN
C      END
C      SUBROUTINE RMREAD(IO,*)
C
C      THIS ROUTINE READS IN R+M FILE IO.
C
C      *****
C
C      DESCRIPTION OF /SHARE/ COMMON BLOCK
C
C      FLIGHTLINE TASKS AND DATA:
C      TSFL - TASK TIME
C      PSM - TASK PROBABILITY
C      SFSE - SUPPORT EQUIPMENT REQUIRED FOR THIS TASK
C      SFAFSC - PERSONNEL REQUIRED FOR THIS TASK
C      NAFSC - NUMBER OF SFAFSC
C      NSFSE - NUMBER OF SFSE
C      FHBMA - MEAN FLIGHT HOURS BETWEEN MAINTENANCE ACTIONS
C      HFAC - HFAC - H FACTOR
C
C      THE SEVEN TASKS ARE:
C      1 - AGE
C      2 - TROUBLESHOOT (TS)
C      3 - CANNOT DUPLICATE (CND)

```

C 4 - REMOVE AND REPLACE (R+R)  
 C 5 - MAINTAIN ON AIRCRAFT (MAC)  
 C 6 - VERIFY R+R (VR+R)  
 C 7 - VERIFY MAC (VMAC)

C\*\*\*\*\*

C DESCRIPTION OF /TEMPRM/ COMMON BLOCK

C FOLLOWING IS DATA ASSOCIATED WITH LRU'S. IN A MANNER SIMILAR  
 C TO THE ABOVE FOR SUBSYSTEMS, TO ALLOW FOR MORE, CHANGE EACH 120 TO THE  
 C DESIRED NUMBER. TO ALLOW FOR MORE AFSC'S PER TASK, CHANGE EACH 3  
 C IN THE LEFTMOST SUBSCRIPT OF LSAFSC TO THE DESIRED NUMBER. TO CHANGE  
 C MAX NUMBER OF SUPPORT EQUIPMENT PER TASK, CHANGE THE 1 IN LSE  
 C TO THE DESIRED NUMBER. CHANGE THE 120, THE 1 AND  
 C THE 3 IN THE FIRST CARD FOLLOWING AND IN THESE COMMENTS.

C SHOP TASKS AND DATA:

C TLSHOP - TASK TIME  
 C PLRR - TASK PROBABILITY  
 C LSAFSC - PERSONNEL REQUIRED FOR THIS TASK  
 C LSE - SUPPORT EQUIPMENT REQUIRED FOR THIS TASK  
 C NLAFSC - NUMBER OF LSAFSC  
 C NLSE - NUMBER OF LSE  
 C WEIGHT - WEIGHT  
 C NSRU - NUMBER OF SRUS

C THE THREE TASKS ARE:

C 1 - SHOP REPAIR (W)  
 C 2 - SHOP CANNOT DUPLICATE (K)  
 C 3 - NRTS (N)

C\*\*\*\*\*

C DESCRIPTION OF /SIZES/ COMMON BLOCK

C NSUB - NUMBER OF SUBSYSTEMS  
 C NLRU - NUMBER OF LRUS  
 C KLRU - STARTING LRU PER SUBSYSTEM  
 C NUML - NUMBER OF LRUS PER SUBSYSTEM  
 C NAF - NUMBER OF AFSC'S  
 C NSE - NUMBER OF SUPPORT EQUIPMENT  
 C NDS - NUMBER OF DEPOT SUPPORT EQUIPMENT (SEE CREAD)  
 C NAI - NUMBER OF AIRCREW (SEE CREAD)

```

C
C*****
C
C   DESCRIPTION OF /EQIDS/ COMMON BLOCK
C
C   SEQID - SUBSYSTEM NAMES
C   LEQID - LRU NAMES
C   AFID - AFSC NAMES
C   SEID - SUPPORT EQUIPMENT NAMES
C
C
C   CHARACTER*7 SFSE,SFAFSC
C   COMMON /SHAREX/ TSFL(7,40),PSH(7,40),SFSE(2,7,40),SFAFSC(5,7,40),
C   *      NSAFSC(7,40),NSFSE(7,40),FHBMA(40),HFAC(40)
C
C   CHARACTER*7 LSAFSC,LSE
C   COMMON /TEMPRM/ TLSHOP(5,120),PLRR(5,120),LSAFSC(5,5,120),
C   *      LSE(2,5,120),NLAFSC(5,120),NLSE(5,120),
C   *      WEIGHT(120),NSRU(120)
C
C   COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
C   CHARACTER*7 AFID,SEID
C   CHARACTER*7 SEQID,LEQID
C   COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)
C
C
C   REAL      TIMES(7),PEAS(7)
C   CHARACTER*1 ITNAME(80),DASH
C   CHARACTER*2 CR,SF,LF,LS,TS,TF,PF,PS,MF,SS
C   CHARACTER*2 TYPE
C   CHARACTER*7 BLANK,DATA(7)
C   CHARACTER*7 EQ
C   CHARACTER*10 ARRAY(8)
C
C   DATA CR,SF,LF,LS,TS,TF,PF,PS,MF,SS
C   & /'CR','SF','LF','LS','TS','TF','PF','PS','MF','SS'/
C   DATA BLANK/' '/
C   DATA MAXLRU,MAXLA,MAXLE/120,5,2/
C   DATA MAXSUB,MAXSA,MAXSE/40,5,2/
C
C READ TITLE CARD
C   REWIND IO
C   READ(IO,9000) ITNAME
C   WRITE(*,9010) ITNAME
C   9000 FORMAT(80A1)
C   9010 FORMAT(/,' TITLE CARD READ: ',/,1X,80A1)
C
C READ NUMBER OF SUBSYSTEMS. HALT IF TOO MANY.
C   NLRU=0

```

```

      READ(10,10) NSUB
10  FORMAT(12)
      IF (NSUB .LE. MAXSUB) GOTO 30
      WRITE(*,20) MAXSUB
20  FORMAT(///,27H CURRENT MAX SUBSYSTEMS AT ,13)
      RETURN 1
C
C READ EACH SUBSYSTEM IN LOOP 100. READ AND WRITE THE CR CARD.
30  DO 100 JSUB=1,NSUB
35  READ(10,40) TYPE,SEQID(JSUB),DASH,JSEQ1,TEMP,NR
40  FORMAT(A2,1X,A7,A1,I1,F6.1,54X,12)
      IF (JSEQ1 .EQ. 2) GOTO 35
      NUML(JSUB)=NR
      IF(JSEQ1 .EQ. 1 .AND. TYPE .EQ. CR) GOTO 40
      GOTO 500
C
C SET POINTER FOR THIS SUBSYSTEM TO FIRST LRU IN LRU TABLES.
60  KLRU(JSUB)=NLRU+1
C
C READ CROSS REFERENCE CARDS FOR EACH LRU IN THIS SUBSYSTEM (LOOP 90).
DO 90 LDUMMY=1,NR
      IF (NLRU .LT. MAXLRU) GOTO 80
      WRITE(*,70) MAXLRU
70  FORMAT(///,21H CURRENT MAX LRUS AT ,13)
      RETURN 1
80  NLRU=NLRU+1
85  READ(10,40) TYPE,LEQID(NLRU),DASH1,JSEQ1,WEIGHT(NLRU),NESS
      IF(NESS .EQ. 0) NESS=1
      NSRU(NLRU)=NESS
      IF(JSEQ1 .EQ. 2) GOTO 85
      IF(JSEQ1 .EQ. 1 .AND. TYPE .EQ. CR) GOTO 90
      GOTO 500
90  CONTINUE
100 CONTINUE
C
C READ MANDATORY SECOND CR CARD FOR LAST LRU
      READ(10,105) ARRAY
105  FORMAT(8A10)
C
C READ SF CARDS. THERE MAY BE FROM 1 TO MAXSE CARDS PER SUBSYSTEM.
      JSUB=0
      DO 150 K=1,NSUB
      READ(10,110) TYPE,EQ,DASH,JSEQ,DATA,NUM
110  FORMAT (A2,1X,A7,A1,I1,7(1X,A5),13)
      IF (TYPE .NE. SF) GOTO 500
      IF (JSEQ .GT. 1) GOTO 500
      IF (NUM .LE. MAXSE) GOTO 118
      WRITE(*,117) MAXSE
117  FORMAT(///,24H CURRENT MAX SE'S SET AT,13)
      GOTO 500
C
C IF CARDS ARE NOT IN SEQUENCE, WE ADVANCE JSUB UP TO THE CORRECT

```

C SUBSYTEM. IF NOT FOUND WE ABORT.

C

```
118 JSUB=JSUB+1
    JFIRST=JSUB
120 IF(EQ .EQ. SEQID(JSUB)) GOTO 130
    JSUB=JSUB+1
    IF(JSUB .GT. NSUB) JSUB=1
    IF (JSUB .NE. JFIRST) GOTO 120
    GOTO 500
```

C

C ASSIGN TO THE PROPER SUBSYSTEM. FIRST 1 SE, THEN THE 2ND, ETC.

C

```
130 NSEQ=1
    DO 135 L=1,7
        NSFSE(L,JSUB)=0
135 CONTINUE
137 DO 140 L=1,7
    IF (DATA(L) .EQ. BLANK) GOTO 140
    NPOS=NSFSE(L,JSUB)+1
    NSFSE(L,JSUB)=NPOS
    SFSE(NPOS,L,JSUB)=DATA(L)
140 CONTINUE
    IF (NSEQ .GE. NUM) GOTO 150
```

C

C READ ADDITIONAL SE'S. THEN STORE ABOVE.

C

```
    NSEQ=NSEQ+1
    READ(IO,110) TYPE,EQ,DASH,JSEQ,DATA
    IF(EQ .NE. SEQID(JSUB) .OR.
      * TYPE .NE. SF .OR.
      * JSEQ .NE. NSEQ) GOTO 500
    GOTO 137
150 CONTINUE
```

C

C READ LF CARDS. THERE MAY BE FROM ONE TO MAXSA CARDS FOR EACH SUBSYSTEM

C

```
    JSUB=0
    DO 200 K=1,NSUB
        READ(IO,110) TYPE,EQ,DASH,JSEQ,DATA,NUM
        IF (TYPE .NE. LF .OR. JSEQ .GT. 1) GOTO 500
        IF (NUM .LE. MAXSA) GOTO 158
        WRITE(*,155) MAXSA
        RETURN 1
155 FORMAT(///,1X,25HCURRENT MAX AFSC'S SET AT,13)
```

C

C IF CARDS ARE NOT IN SEQUENCE BY SUBSYSTEM, WE INCREMENT JSUB TO IT.

C

```
158 JSUB=JSUB+1
    JFIRST=JSUB
160 IF (EQ .EQ. SEQID(JSUB)) GOTO 170
    JSUB=JSUB+1
    IF(JSUB .GT. NSUB) JSUB=1
```

```

        IF (JSUB .NE. JFIRST) GOTO 160
        GOTO 500
C
C ASSIGN TO PROPER SUBSYSTEM, INITIALLY THE FIRST AFSC, THEN SECOND, ETC
C
170 NSEQ=1
    DO 173 L=1,7
        NSAFSC(L,JSUB)=0
173 CONTINUE
175 DO 180 L=1,7
    IF (DATA(L) .EQ. BLANK) GOTO 180
    NPOS=NSAFSC(L,JSUB)+1
    NSAFSC(L,JSUB)=NPOS
    SFAFSC(NPOS,L,JSUB)=DATA(L)
180 CONTINUE
    IF (NSEQ .GE. NUM) GOTO 200
C
C READ ADDITIONAL AFSC'S, THEN STORE SIMILARLY ABOVE.
C
    NSEQ=NSEQ+1
    READ(10,110) TYPE,EQ,DASH,JSEQ,DATA
    IF (EQ .NE. SEQID(JSUB) .OR.
*     TYPE .NE. LF .OR.
*     JSEQ .NE. NSEQ) GOTO 500
    GOTO 175
200 CONTINUE
C
C READ LS CARDS. THERE MAY BE FROM ONE TO MAXLA CARDS PER LRU.
C
    JLRU=0
    DO 260 K=1,NLRU
        READ(10,210) TYPE,EQ,DASH,JSEQ,DATA,NUM
210 FORMAT(A2,1X,A7,A1,I1,6X,7(1X,A5),13)
        IF (TYPE .NE. LS .OR. JSEQ .GT. 1) GOTO 500
        IF (NUM .LE. MAXLA) GOTO 215
        WRITE(*,135) MAXLA
        GOTO 500
215 JLRU=JLRU+1
        JFIRST=JLRU
220 IF (EQ .EQ. SEQID(JLRU)) GOTO 230
C
C IF CARDS ARE NOT IN SEQUENCE BY LRU, WE INCREMENT JLRU UP TO IT.
C
    JLRU=JLRU+1
    IF (JLRU .GT. NLRU) JLRU=1
    IF (JLRU .EQ. JFIRST) GOTO 500
    GOTO 220
C
C ASSIGN TO PROPER LRU
C
230 NSEQ=1

```

```

      DO 235 L=1,5
      NLAFSC(L,JLRU)=0
235  CONTINUE
240  DO 250 L=1,5
      LX=L
      IF (L .GT. 3) LX=L+2
      IF (DATA(LX) .EQ. BLANK) GOTO 250
      NPOS=NLAFSC(L,JLRU)+1
      NLAFSC(L,JLRU)=NPOS
      LSAFSC(NPOS,L,JLRU)=DATA(LX)
250  CONTINUE
      IF (NSEQ .GE. NUM) GOTO 260
C
C READ ADDITIONAL AFSC'S, THEN STORE SIMILARLY ABOVE.
C
      NSEQ=NSEQ+1
      READ(10,210) TYPE,EQ,DASH,JSEQ,DATA
      IF(EQ .NE. LEQID(JLRU)) .OR.
*   TYPE .NE. LS .OR.
*   JSEQ .NE. NSEQ) GOTO 500
      GOTO 240
260  CONTINUE
C
C READ TS CARDS, ONE PER LRU IN ANY ORDER
C
      JLRU=0
      DO 300 K=1,NLRU
      READ(10,265) TYPE,EQ,DASH,JSEQ,TIMES
265  FORMAT(A2,1X,A7,A1,11,6X,7(1X,F5.1))
      IF (TYPE .NE. TS .OR. JSEQ .GT. 1) GOTO 500
C
C IF CARDS ARE NOT IN SEQUENCE BY LRU, WE INCREMENT JLRU UP TO IT.
C
      JLRU=JLRU+1
      JFIRST=JLRU
270  IF (EQ .EQ. LEQID(JLRU)) GOTO 280
      JLRU=JLRU+1
      IF (JLRU .GT. NLRU) JLRU=1
      IF (JLRU .NE. JFIRST) GOTO 270
      GOTO 500
C
C ASSIGN TO PROPER LRU
C
280  DO 290 L=1,3
      TLSHOP(L,JLRU)=TIMES(L)
290  CONTINUE
      TLSHOP(4,JLRU)=TIMES(6)
      TLSHOP(5,JLRU)=TIMES(7)
300  CONTINUE
C
C READ TF CARDS, ONE PER SUBSYSTEM
C

```

```

JSUB=0
DO 340 K=1,NSUB
  READ(10,305) TYPE,EQ,DASH,JSEQ,TIMES
305 FORMAT(A2,1X,A7,A1,I1,7(1X,F5.1))
  IF (TYPE .NE. TF .OR. JSEQ .GT. 1) GOTO 500
C
C IF CARDS ARE NOT IN SEQUENCE, WE INCREMENT JSUB UP TO IT.
C
  JSUB=JSUB+1
  JFIRST=JSUB
310 IF (EQ .EQ. SEQID(JSUB)) GOTO 320
  JSUB=JSUB+1
  IF (JSUB .GT. NSUB) JSUB=1
  IF (JSUB .NE. JFIRST) GOTO 310
  GOTO 500
C
C ASSIGN TO PROPER SUBSYSTEM
C
320 DO 330 L=1,7
  TSFL(L,JSUB)=TIMES(L)
330 CONTINUE
340 CONTINUE
C
C READ PF CARDS, ONE PER SUBSYSTEM.
C
  JSUB=0
  DO 390 K=1,NSUB
    READ(10,350) TYPE,EQ,DASH,JSEQ,PEAS
350 FORMAT(A2,1X,A7,A1,I1,7(F6.4))
    IF (TYPE .NE. PF .OR. JSEQ .GT. 1) GOTO 500
C
C IF CARDS ARE NOT IN SEQUENCE, WE INCREMENT JSUB UP TO IT.
C
  JSUB=JSUB+1
  JFIRST=JSUB
360 IF (EQ .EQ. SEQID(JSUB)) GOTO 370
  JSUB=JSUB+1
  IF (JSUB .GT. NSUB) JSUB=1
  IF (JSUB .NE. JFIRST) GOTO 360
  GOTO 500
C
C ASSIGN TO PROPER SUBSYSTEM
C
370 DO 380 L=1,7
  PSM(L,JSUB)=PEAS(L)
380 CONTINUE
390 CONTINUE
C
C READ PS CARDS, ONE PER LRU IN ANY ORDER
C
  JLRU=0
  DO 440 K=1,NLRU

```

```

      READ(10,400) TYPE,EQ,DASH,JSEQ,PEAS
400  FORMAT(A2,1X,A7,A1,I1,6X,7(F6.4))
      IF (TYPE .NE. PS .OR. JSEQ .GT. 1) GOTO 500
C
C IF CARDS ARE NOT IN SEQUENCE, WE INCREMENT JLRU TO IT.
C
      JLRU=JLRU+1
      JFIRST=JLRU
410  IF (EQ .EQ. LEQID(JLRU)) GOTO 420
      JLRU=JLRU+1
      IF (JLRU .GT. NLRU) JLRU=1
      IF (JLRU .NE. JFIRST) GOTO 410
      GOTO 500
C
C ASSIGN TO PROPER LRU
C
420  DO 430  L=1,3
      PLRR(L,JLRU)=PEAS(L)
430  CONTINUE
      PLRR(4,JLRU)=PEAS(6)
      PLRR(5,JLRU)=PEAS(7)
440  CONTINUE
C
C READ 88 CARDS, ONE PER LRU.  ADDITIONAL SE'S ON FOLLOWING CARDS.
C
      JLRU=0
      DO 449  K=1,NLRU
      READ(10,441) TYPE,EQ,DASH,JSEQ,(DATA(J),J=1,3),DATA(4),
&          DATA(5),NUM
441  FORMAT(A2,1X,A7,A1,I1,6X,3(1X,A5),12X,2(1X,A5),1X,I2)
      IF (TYPE .NE. SS .OR. JSEQ .GT. 1) GOTO 500
      IF (NUM .LE. MAXLE) GOTO 4410
      WRITE(*,117) MAXLE
      RETURN 1
4410  JLRU=JLRU+1
      JFIRST=JLRU
442  IF (EQ .EQ. LEQID(JLRU)) GOTO 443
C
C IF CARDS ARE NOT IN SEQUENCE, WE INCREMENT JLRU UP TO IT.
C
      JLRU=JLRU+1
      IF (JLRU .GT. NLRU) JLRU=1
      IF (JLRU .EQ. JFIRST) GOTO 500
      GOTO 442
C
C ASSIGN TO PROPER LRU
C
443  NSEQ=1
      DO 444  L=1,5
      NLSE(L,JLRU)=0
444  CONTINUE
445  DO 446  L=1,5

```

```

      IF (DATA(L) .EQ. BLANK) GOTO 446
      NPOS=NLSE(L,JLRU)+1
      NLSE(L,JLRU)=NPOS
      LSE(NPOS,L,JLRU)=DATA(L)
446  CONTINUE
      IF (NSEQ .GE. NUM) GOTO 449
C
C  READ ADDITIONAL SE'S, THEN STORE ABOVE.
C
      NSEQ=NSEQ+1
      READ(10,441) TYPE,EQ,DASH,JSEQ,(DATA(J),J=1,3),DATA(4),DATA(5)
      IF(EQ .NE. LEQID(JLRU)) .OR.
      *   TYPE .NE. SS .OR.
      *   JSEQ .NE. NSEQ) GOTO 500
      GOTO 445
449  CONTINUE
C
C  READ MF CARDS, 1 PER SUBSYSTEM IN ANY ORDER
C
      JSUB=0
      DO 480 K=1,NSUB
      READ(10,450) TYPE,EQ,DASH,JSEQ,VAL,H
450  FORMAT(A2,1X,A7,A1,I1,1X,F6.1,1X,F6.4)
      IF (TYPE .NE. MF .OR. JSEQ .GT. 1) GOTO 500
C
C  IF CARDS ARE NOT IN SEQUENCE, INCREMENT JSUB UP TO IT.
C
      JSUB=JSUB+1
      JFIRST=JSUB
460  IF (EQ .EQ. SEQID(JSUB)) GOTO 470
      JSUB=JSUB+1
      IF (JSUB .GT. NSUB) JSUB=1
      IF (JSUB .NE. JFIRST) GOTO 460
      GOTO 500
C
C  ASSIGN TO PROPER SUBSYSTEM
C
      470  FHBMA(JSUB)=VAL
      HFAC(JSUB)=H+1.0
      480  CONTINUE
      WRITE(*,9020)
9020  FORMAT(/,' FINISHED READING R&M DATA.')
      RETURN
C
C  PRINT ERROR MESSAGE AND ABNORMALLY RETURN
C
      500  WRITE(*,510)
      510  FORMAT(///,' R+M INPUT FILE ERROR -- USER NEEDS TO FIX IT')
      WRITE(*,515) TYPE,EQ,JSEQ
      515  FORMAT(/,1X,A2,1X,A7,I2)
      RETURN 1
      END

```

```

      SUBROUTINE CMODEL
C
C   THIS IS THE DRIVER FOR THE COST CALCULATIONS.
C       IO3 - COST INPUT FILE
C       JN - 0 IF NO R+M PERTURBED DATA
C           - 1 IF THERE IS PERTURBED DATA
C       IO3A - BASE OUTPUT FILE (RETURNED)
C       IO4A - PERTURBED OUTPUT FILE (RETURNED)
C
C   CALLED BY MAIN
C
C   IOIN - COST DATA FILE UNIT NUMBER
C   IOOUT - CURRENT OUTPUT FILE UNIT NUMBER
C   NTH - 1 FOR BASE R+M DATA
C         - 2 FOR PERTURBED R+M DATA FROM /RAM/
C
C
C       COMMON /OVER/ JABT,IO2,IO4A,JN,NMAX,LAST
C
C       INTEGER MASK,TITLE
C       COMMON /ALL/ NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
C       COMMON /COSTIO/ IOIN,IOOUT,NTH
C
C   GET BASE OUTPUT FILE, WRITE FIRST LINE AND READ COST INPUTS.
C
C       CALL CHECK(13,*1)
C       GOTO 2
C   1 WRITE(*,6)
C   6 FORMAT(///, ' COST FILE IS NOT ATTACHED')
C       GOTO 60
C   2 CALL CHECK(-14,*4)
C       CALL CHECK(-5,*4)
C       GOTO 7
C   4 WRITE(*,8)
C   8 FORMAT(///, ' INTERNAL ERROR - CMODEL')
C       STOP
C   7 NTH=1
C       IOIN=13
C       IOOUT=14
C       CALL OUTFIL(1,*60)
C       CALL CREAD(*60)
C
C   ALLOW MODIFICATION OF COST INPUTS. THEN CALCULATE RESULTS OF
C   THE MODEL FOR THE BASE CASE.
C
C   3 IF (NOTHER .EQ. 0) WRITE(*,5)
C   5 FORMAT(/, ' DO YOU WANT TO CHANGE INITIAL COST INPUTS? ')
C       CALL INP(3,0,0,0,0,240,IC,*60,*3)
C       IF (IC .EQ. 1) CALL MODCST(IO3,1)
C       CALL CALCOS
C

```

```

C ADD MMH FILE TO OUT3
C
    CALL MMHCOP(4,14)
C
C SET NTH TO 2 FOR PERTURBED R+M DATA. ALLOW PERTURBATION OF
C COST DATA.
C
    IO4A=0
    NTH=JN+1
    ICOP = 4
    IF (JN .EQ. 1) ICOP = 7
10 IF (NOTHER .EQ. 0) WRITE(*,20)
20 FORMAT(/, ' DO YOU WANT TO PERTURB COSTS?      ')
    CALL INP(3,0,0,0,0,250,JC,*60,*10)
    IF (JC .EQ. 0) IF (JN) 58,58,35
    CALL MODCST(103,2)
C
C IF R+M OR COST OR BOTH ARE PERTURBED, GET OUTPUT FILE, WRITE
C THE FIRST LINE, AND RE-CALCULATE COSTS.
C
35 CALL CHECK(-15,*4)
    CALL CHECK(-6,*4)
    IOUT=15
    IO4A=15
    CALL OUTFIL(2,*60)
    CALL CALCOS
C
C ADD MMH FILE TO IO4A
C
    CALL MMHCOP(ICOP,15)
58 RETURN
60 JABT=1
    RETURN
    END
    SUBROUTINE MMHCOP(IN,IO)
    REWIND IN
5 READ(IN,END=20) I,J,X
    WRITE(IO) I,J,X
    WRITE(IO-9,99) I,J,X
99 FORMAT(I10,2X,I10,2X,E15.8)
    BOTO 5
20 WRITE(IO) 0, 0, 0.0
    WRITE(IO-9,99) 0, 0, 0.0
    RETURN
    END
    SUBROUTINE CREAD(*)
C
C THIS ROUTINE READS COST INPUT DATA TO /SHARE/.
C
C CALLED BY CMODEL
C
C

```

```

C CHARACTER*7 AFID,SEID
C CHARACTER*7 SEQID,LEQID
COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)
C
C COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
C COMMON /COSTIO/ IOIN,IOOUT,NTH
C
C COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
C COMMON /ERR/ JERR,KOUNT
C
C CHARACTER*8 FIELDS
COMMON /EX/ FIELDS(8)
C
C CHARACTER*1 BL,DA,DASH,BLANK,INTITLE(80)
C CHARACTER*2 CT,ID(8)
C CHARACTER*7 NAME
C INTEGER LVE1(4), LVE2(4)
C INTEGER LVS1(10),LVS2(10)
C INTEGER LVN1(2), LVN2(2)
C INTEGER LVI1(2), LVI2(2)
C INTEGER LVM1, LVM2
C INTEGER LVJ1(2), LVJ2(2)
C INTEGER LVD1, LVD2
C REAL TDATA(15)
C DIMENSION X(3175)
C EQUIVALENCE (X(1),SIN(1,1))
C
C DATA DASH,BLANK/'-',' ' /
C DATA ID/'VE','VS','VN','VI','VM','VJ','VP','VD' /
C
C DATA LVE1 /1,9,14,22/
C DATA LVE2 /8,13,21,27/
C
C DATA LVS1 /1,9,17,25,33,41,49,57,65,73/
C DATA LVS2 /8,16,24,32,40,48,56,64,72,78/
C
C DATA LVN1 /1,9/
C DATA LVN2 /8,13/
C
C DATA LVI1 /1,9/
C DATA LVI2 /8,13/
C
C DATA LVM1 /1/
C DATA LVM2 /4/
C
C DATA LVJ1 /1,9/
C DATA LVJ2 /8,9/
C
C DATA LVD1 /1/

```

**١٢**

[illegible]

C  
C  
C  
C

C  
C

C  
C  
C

C  
C

C  
C  
C

CC

CCC

CC CC

C  
C  
  
C  
C  
C

CC

CCC

CC

CCC

CC CC

CC CC

**C C C**

**C  
C  
C**

**C  
C  
C**

**C  
C  
C**

**C  
C  
C**

**CCC**

**C**

```

50 CALL ABORT(KOUNT,*90)
   WRITE(*,60)
60 FORMAT(/,' CARD FORMAT ERROR IN THE FOLLOWING CARD...')
70 WRITE(*,80) CT,BL,NAME,DA,JSEQ,FIELDS
80 FORMAT(1X,A2,A1,A7,A1,I1,3X,8A8)
90 JERR=JERR+1
   NOTHER=0
   BOTO 10

C
C IN THE FOLLOWING SEGMENTS, L__1 AND L__2 FOR THE CARD TYPE
C TELL EXTRAK WHERE IN THE CALLING ARRAY TO STORE THE DATA
C READ FROM THE CARD. ALL CARDS WITH SEQUENCE NUMBERS OUT OF
C RANGE ARE REPORTED AT 110. ALL CARDS WITH NAMES NOT FOUND
C IN R+M DATA ARE REPORTED AT 322.
C
C
C VE
C
100 IF (JSEQ .GE. 1 .AND. JSEQ .LE. 4) BOTO 130
110 CALL ABORT(KOUNT,*90)
   WRITE(*,120)
120 FORMAT(/,' THE FOLLOWING CARD HAS AN INVALID SEQUENCE NUMBER...')
   BOTO 70
130 CALL EXTRAK(LVE1(JSEQ),LVE2(JSEQ),COSTS)
   BOTO 10

C
C VS
C
200 IF (JSEQ .EQ. 0) JSEQ=10
   CALL EXTRAK(LVS1(JSEQ),LVS2(JSEQ),SCAL)
   BOTO 10

C
C VN
C
300 DO 320 N=1,NAF
   IF (NAME .EQ. AFID(N)) BOTO 330
320 CONTINUE
322 CALL ABORT(KOUNT,*90)
   WRITE(*,325) NAME
325 FORMAT(/,1X,A7,' NOT IN SYSTEM.')
   BOTO 90

C
C
330 IF (JSEQ .LT. 1 .OR. JSEQ .GT. 2) BOTO 110
   L1=LVN1(JSEQ)
   L2=LVN2(JSEQ)
   CALL EXTRAK(L1,L2,TDATA)
   DO 340 L=L1,L2
   AFIN(N,L) = TDATA(L)
340 CONTINUE
   BOTO 10

C

```

```

C VI
C
400 DO 410 I=1,NLRU
    IF (NAME .EQ. LEQID(I)) GOTO 420
410 CONTINUE
    GOTO 322
C
420 IF (JSEQ.LT.1.OR.JSEQ.GT.2) GOTO 110
    L1=LV11(JSEQ)
    L2=LV12(JSEQ)
    CALL EXTRAK(L1,L2,TDATA)
    DO 430 L=L1,L2
        RUIN(I,L)=TDATA(L)
430 CONTINUE
    GOTO 10
C
C VM
C
500 DO 510 M=1,NSUB
    IF (NAME .EQ. SEQID(M)) GOTO 520
510 CONTINUE
    GOTO 322
C
520 IF (JSEQ .NE. 1) GOTO 110
    CALL EXTRAK(LVM1,LVM2,TDATA)
    DO 530 L=LVM1,LVM2
        SIN(M,L) = TDATA(L)
530 CONTINUE
    GOTO 10
C
C VJ
C
600 DO 610 J=1,NSE
    IF (NAME .EQ. SEID(J)) GOTO 620
610 CONTINUE
    GOTO 322
C
620 IF (JSEQ .LT. 1 .OR. JSEQ .GT. 2) GOTO 110
    L1 = LVJ1(JSEQ)
    L2 = LVJ2(JSEQ)
    CALL EXTRAK(L1,L2,TDATA)
    DO 630 L=L1,L2
        SEIN(J,L)=TDATA(L)
630 CONTINUE
    GOTO 10
C
C VP
C THE PRESENCE OF A VP CARD INFERS ONE MORE AIRCREW TYPE.
C
700 IF (JSEQ .NE. 1) GOTO 110
    NAI=NAI+1
    IF (NAI .GT. 50) GOTO 710

```

```

      CALL EXTRAK(1,1,AIDATA(NAI))
      GOTO 10
710  WRITE(*,711)
711  FORMAT(/,' NUMBER OF AIRCREW EXCEEDS 50.')
```

NOTHER=0  
 GOTO 5

C  
 C VD  
 C THE PRESENCE OF A VD CARD INFERS ONE MORE DEPOT SUPPORT CARD.  
 C

```

800  IF (JSEQ .NE. 1) GOTO 110
      NDS = NDS + 1
      IF (NDS .GT. 50) GOTO 810
      CALL EXTRAK (LVD1,LVD2,TDATA)
      DSE(NDS,1) = TDATA(1)
      DSE(NDS,2) = TDATA(2)
      GOTO 10
810  WRITE(*,811)
811  FORMAT(/,' NUMBER OF DEPOT SUPPORT EQUIPMENT EXCEEDS 50.')
```

NOTHER=0  
 GOTO 5

C

```

890  IF (JERR .EQ. 0) GOTO 900
      WRITE(*,895) JERR
895  FORMAT(/,' EXECUTION HALTED.',14,' ERRORS.')
```

5 RETURN 1

C

```

900  WRITE(*,9050)
9050 FORMAT(/,' FINISHED READING COST DATA.')
```

RETURN  
 END  
 SUBROUTINE EXTRAK(L1,L2,TDATA)

C  
 C THIS ROUTINE DECODES VALUES FROM FIELDS (IN COMMON) AND  
 C STORES THEM IN ARRAY TDATA FROM POSITION L1 TO L2. NOTE  
 C THE NUMBER OF DATA ITEMS DECODED IS 1 + L2 - L1.  
 C

```

      REAL TDATA(1)
      CHARACTER*10 BLANKS
    
```

C

```

      CHARACTER*8 FIELDS,TEMP
      COMMON /EX/ FIELDS(8)
    
```

C

```

      DATA BLANKS/' '
    
```

C  
 C  
 C  
 C KEEP FLAG OF LARGE NEGATIVE NUMBER SET IN CASE ONE OF THE DATA  
 C ITEMS IS BLANK. HENCE BLANK IS NOT ZERO.  
 C

```

      K=0
      DO 20 L=L1,L2
        VAL = -1.E30
    
```

```

      K=K+1
      TEMP=FIELDS(K)
      IF (TEMP .NE. BLANKS) READ(TEMP,10) VAL
10  FORMAT(F8.0)
      TDATA(L) = VAL
20  CONTINUE
      RETURN
      END
      SUBROUTINE ERROR(VAR)
C
C  OUTPUTS A MESSAGE INDICATING ABSENCE OF DATA NECESSARY FOR A
C  GIVEN COST RESULT (VAR).
C
      CHARACTER*6 VAR
      COMMON /ERR/ JERR,KOUNT
      CALL ABORT(KOUNT,*20)
      WRITE(*,10) VAR
10  FORMAT(/,' INSUFFICIENT DATA TO COMPUTE ',A6)
      JERR=JERR+1
20  RETURN
      END
      SUBROUTINE CALCOS
C
C  THIS SUBROUTINE INITIATES CALCULATION OF MODEL OUTPUTS.  THE
C  INPUTS ARE COMMON /RAM/ AND COMMON /SHARE/.  THE OUTPUTS ARE
C  WRITTEN ONTO FILE IQUT.  EACH OUTPUT HAS A PRINT CODE AS
C  DESCRIBED IN 'OUTPUT'.  EACH IS WRITTEN BY A CALL TO RITE.
C  IF AN OUTPUT IS AN ARRAY WHICH VARIES WITH A SUBSYSTEM, AFSC,
C  ETC., THERE IS 1 LINE PER SUBSYSTEM, AFSC, ETC.
C
C  SUBROUTINES TO COMPUTE A COST (OR ANY OUTPUT) ARE NAMED BY
C  THE LETTER S FOLLOWED BY THE COST.  SCO COMPUTES CO.
C
C  ALL COSTS ARE COMPUTED TO THE LOWEST LEVEL, WITH THE FOLLOWING
C  EXCEPTIONS:
C    1) FOR LRU'S, UCSRU, IC, AND CALI ARE USED DIRECTLY IF
C       INPUT INSTEAD OF CALCULATING LOWER LEVEL EQUATIONS.
C       IF UNAVAILABLE ON INPUT, THE ATTEMPT WILL BE MADE TO
C       CALCULATE FROM LOWER LEVEL DATA.
C    2) IF ANY OR ALL OF THE LOWER LEVEL DATA OF A COST RESULT
C       IS NOT AVAILABLE ON INPUT, THAT RESULT WILL BE TAKEN
C       DIRECTLY FROM THE HIGH LEVEL INPUT ITSELF (IF UNAVAILABLE
C       ALSO, AN ERROR).
C
C  THE ABSENCE OF DATA IS SIGNIFIED BY -10**30 STORED IN THE
C  VARIABLE.  IF SO, COSTS FOR WHICH IT IS A PART MUST BE TAKEN
C  FROM HIGHER LEVEL DATA.
C
C  ALL VARIABLE NAMES SIMILAR TO THE ALGEBRAIC EQUATIONS ARE
C  LOCAL WITH THE EXCEPTION OF /BASIC/.  ALL NAMES ARE INVENTED
C  SO AS TO PHONETICALLY SOUND LIKE THE EQUATION NAME YET STILL
C  RETAIN THE REAL/INTEGER FIRST LETTER CONVENTION.  (E.G. ENRC

```

```

C   FOR NRC, OR EYEC FOR IC)
C
C   CONDITIONAL RETURNS FROM A SUBROUTINE INFER THE DATA WAS NOT
C   AVAILABLE FOR COMPUTATION.
C
C   CALLED BY CREAD
C
C   THESE ARE VARIABLES USED FREQUENTLY IN THE COST EQUATIONS:
C
C   VARIABLE      MNEMONIC
C   -----
C   ABFH          ABFH
C   NACB          NACB
C   PBFH          PBFH
C   ENNII         ENNII
C
C   COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
C   *              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
C   COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
C   COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUHL(40),NAF,NSE,NDS,NAI
C
C   COMMON /COSTIO/ IOIN,IOUT,NTH
C
C   WRITE(IOUT)      NSUB,NLRU,NAF,NSE,NAI,NDS
C   WRITE(IOUT-9,999) NSUB,NLRU,NAF,NSE,NAI,NDS
999 FORMAT(6I10)
C   ENNII = 0.0
C   DO 5 I=1,NLRU
C   PA = RUIN(I,9)
C   IF (PA .LE. -1.E30) GOTO 7
C   PP = RUIN(I,10)
C   IF (PP .LE. -1.E30) GOTO 7
C   ENNII = ENNII + 1.0 + PA + PP
C   5 CONTINUE
C   GOTO 8
C   7 ENNII = -1.E30
C   8 EM = SCAL(47)
C   ABFH = -1.E30
C   PBFH = -1.E30
C   NACB = SCAL(49)
C   IF (NACB .LE. -1.E30) GOTO 20
C   FHACH = SCAL(52)
C   IF (FHACH .LE. -1.E30) GOTO 10
C   PBFH = NACB * FHACH * 12.0
C   10 AFHACH = SCAL(51)
C   IF (AFHACH .LE. -1.E30) GOTO 20
C   ABFH = NACB * AFHACH * 12.0

```

```

C
C THE FOLLOWING TWO SUBROUTINES STORE THE RESULTS DIRECTLY INTO
C THE /SHARE/ ARRAYS.
C
C 20 CALL SCJBI
C    CALL SCPUSE
C
C NON-RECURRING COSTS
C
C    CRD = COSTS(14)
C    IF (CRD .LE. -1.E30) CALL ERROR('CRD  ')
C    CALL SCBI(CSI)
C    CALL SDOI(COI)
C    ENRC = CRD + CSI + COI
C
C RECURRING COSTS
C
C    CALL SCO(CO)
C    CALL SCS(CS)
C    RCY = CO + CS
C
C
C    CDP = COSTS(27)
C    IF (CDP .LE. -1.E30) CALL ERROR('CDP  ')
C    PIUP = SCAL(48)
C    RC = RCY + PIUP
C    ELCC = ENRC + RC + CDP
C    CALL RITE(52,CRD)
C    CALL RITE(53,ENRC)
C    CALL RITE(54,RC)
C    CALL RITE(55,CDP)
C    CALL RITE(56,ELCC)
C    CALL RITE(58,RCY)
C
C ADJUSTED COSTS
C
C    CALL LCCADJ(ENRC,CDP,RCY)
C    CALL EXAM
C    RETURN
C    END
C    SUBROUTINE LCCADJ(ENRC,CDP,RCY)
C
C THIS ROUTINE CALCULATES LCC ADJUSTED FOR CHANGE IN INFLATION
C RATE AND DISCOUNT RATE.
C
C    ENRC - NON-RECURRING COSTS
C    CDP - DISPOSAL COSTS
C    RCY - ANNUAL RECURRING COSTS
C
C CALLED BY CALCOS
C
C    COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),

```

```

      *                DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
C
C CAN'T DO IT WITHOUT ALL THE INPUT.
C
      IF (SCAL(65) .LE. -1.E30) RETURN
      EYER=SCAL(65)
      IF (SCAL(72) .LE. -1.E30) RETURN
      DR=SCAL(72)
      IF (SCAL(71) .LE. -1.E30) RETURN
      IDPT=SCAL(71)
      IF (SCAL(70) .LE. -1.E30) RETURN
      IPOT=SCAL(70)
      IF (SCAL(48) .LE. -1.E30) RETURN
      IPIUP=SCAL(48)
      TEMP=(1.0+EYER)/(1.0+DR)
C
C BREAK UP NON-RECURRING COSTS OVER THE TIME PERIOD AND MODIFY
C EACH YEAR BY APPLICABLE FACTOR.
C
      HOLD=ENRC/IDPT
      ENRCT=0.0
      DO 20 I=1,IDPT
      IT=I-1
      TENRCT=AAF(TEMP,IT)*HOLD
      CALL RITE(80,TENRCT)
      ENRCT=ENRCT+TENRCT
20 CONTINUE
C
C SPREAD ANNUAL RECURRING COSTS OVER THE USAGE PERIOD. TIME
C PERIOD BEGINS AFTER IDPT.
C
      RCYT=0.0
      DO 30 I=1,IPIUP
      IT=IDPT+I-1
      TRCYT=AAF(TEMP,IT)*RCYT
      CALL RITE(81,TRCYT)
      RCYT=RCYT+TRCYT
30 CONTINUE
C
C BREAK UP DISPOSAL COSTS OVER THE TIME PERIOD AND MODIFY EACH
C YEAR BY APPLICABLE FACTOR. TIME PERIOD BEGINS AFTER IDPT AND
C IPIUP.
C
      IAFTER=IDPT+IPIUP
      CDPT=0.0
      IF (IPOT .LE. 0.0) GOTO 45
      SEMI=CDP/IPOT
      DO 40 I=1,IPOT
      IT=IAFTER+I-1
      TCDPT=AAF(TEMP,IT)*SEMI
      CALL RITE(82,TCDPT)

```

```

      CDPT=CDPT+TCDPT
40  CONTINUE
C
C 45  ELCCAJ=ENRCT+RCYT+CDPT
      CALL RITE(83,ELCCAJ)
      RETURN
      END
      FUNCTION AAF(TEMP,IT)
C
C THIS FUNCTION CONVERTS INFLATION RATE DIVIDED BY DISCOUNT
C RATE (TEMP) AND NUMBER OF YEARS HENCE (IT) TO A DOLLAR FUDGE
C FACTOR.
C
C CALLED BY LCCADJ
      IF (IT .EQ. 0) GOTO 10
      AAF=(TEMP**((IT-1)+TEMP**IT))/2.0
      RETURN
10  AAF=1.0
      RETURN
      END
      SUBROUTINE MODTYP(OLD,VAL,KP,GNU)
C
C CALLED BY MODCST TO MODIFY OLD, BY MEANS OF VAL, USING TYPE
C KP,AND RETURNING NEW.
C
C KP = 1 - BIAS
C      2 - FACTOR
C      3 - REPLACE
C
      IF (KP-2) 10,20,30
10  GNU=OLD+VAL
      RETURN
20  GNU=OLD*VAL
      RETURN
30  GNU=VAL
      RETURN
      END
      SUBROUTINE OUTFIL(JFLAG,*)
C
C THIS ROUTINE IS CALLED BY CMODEL TO WRITE THE FIRST RECORD
C OF THE OUTPUT FILE, WHICH IS SIMPLY THE FIRST RECORD OF
C THE INPUT FILE PLUS THE WORD 'OUTPUT' PLUS (IF A PERTURBED
C OUTPUT) AN OPTIONAL TITLE.
C
C JFLAG = 1 - REGULAR OUTPUT FILE
C      2 - PERTURBED OUTPUT FILE
C
C CHARACTER*1 BL,BLANK(10)
C CHARACTER*10 OUTS,HEAD(6)
C
C COMMON/COSTIO/IOIN,IOUT,NTH
C

```

```

COMMON/ALL/NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
CHARACTER*1 XMASK,XTITLE
COMMON /JJF/ XMASK(10),XTITLE(10)
C
DATA OUTS/' OUTPUT '/
DATA BL/' '/
C
DO 5 N=1,10
  BLANK(N) = BL
5 CONTINUE
C
REWIND IOIN
READ(IOIN,10,END=50) HEAD
10 FORMAT(6A10)
REWIND IOUT
IF (JFLAG .EQ. 1) GOTO 30
C
C GET TITLE
C
IF (NTITL .EQ. -1) GOTO 20
NT=NTITL
GOTO 40
20 IF (NOTHER .EQ. 0) WRITE(*,21)
21 FORMAT(/,' PERTURBED OUTPUT FILE TITLE? ')
CALL INP(5,0,XTITLE,0,10,180,NT,*50,*20)
GOTO 40
C
30 WRITE(IOUT) HEAD,OUTS,BLANK
WRITE(IOUT-9,99) HEAD,OUTS,BLANK
99 FORMAT(6A10,A10,10A1)
RETURN
C
40 WRITE(IOUT) HEAD,OUTS,XTITLE
WRITE(IOUT-9,99) HEAD,OUTS,XTITLE
RETURN
C
50 RETURN 1
END
SUBROUTINE MODCST(103,IJ)

```

```

C
C MODCST IS CALLED BY CMODEL TO MODIFY THE COST INPUT IN ARRAYS
C IN /RAM/ AND /SHARE/. THE VARIABLES WHICH CAN BE MODIFIED,
C ALONG WITH THE ARRAY IN WHICH EACH IS FOUND, AND THE POSITION
C WITHIN THE ARRAY IS SHOWN BELOW. ALSO SHOWN IS THE PRINT
C FORMAT USED:

```

K	VARIABLE	WHICH ARRAY	WHERE IN ARRAY	PRINT FORMAT
1	NB	11	SCAL	47 2
2	MFHBMA	1	SDAT	8 2
3	SNSRU	1	SDAT	9 1

C	4	CPINT	2	SIN	1	2
C	5	CINST	2	SIN	2	2
C	6	CFJ8	2	SIN	3	2
C	7	CSJ8	2	SIN	4	2
C	8	W	3	UDAT	3	2
C	9	PM	3	UDAT	4	3
C	10	PH	3	UDAT	5	3
C	11	PS	3	UDAT	6	3
C	12	LNSRU	3	UDAT	7	1
C	13	UC	4	RUIN	1	2
C	14	UCSRU	4	RUIN	2	2
C	15	FC	11	SCAL	53	2
C	16	FCS	4	RUIN	4	2
C	17	T	4	RUIN	5	2
C	18	DR	11	SCAL	72	2
C	19	DC	4	RUIN	7	2
C	20	TC	4	RUIN	8	2
C	21	PA	4	RUIN	9	1
C	22	PP	4	RUIN	10	1
C	23	SP	4	RUIN	11	1
C	24	IC	4	RUIN	12	2
C	25	CALI	4	RUIN	13	2
C	26	FNMH	5	ADAT	1	2
C	27	SNMH	5	ADAT	2	2
C	28	NW	11	SCAL	17	1
C	29	ACB	6	AFIN	2	2
C	30	CIC	6	AFIN	3	2
C	31	COT	6	AFIN	4	2
C	32	CTTS	6	AFIN	5	2
C	33	COJT	6	AFIN	6	2
C	34	TCB	6	AFIN	7	2
C	35	TRS	6	AFIN	8	2
C	36	CHPS	6	AFIN	9	2
C	37	OPF	6	AFIN	10	2
C	38	KH	6	AFIN	11	2
C	39	DLR	6	AFIN	12	2
C	40	LLR	6	AFIN	13	2
C	41	BMR	6	AFIN	14	2
C	42	TSDEM	7	EDAT	1	4
C	43	TSDOT	7	EDAT	2	4
C	44	UCSE	8	SEIN	1	2
C	45	KI	11	SCAL	32	2
C	46	KSE	8	SEIN	3	2
C	47	CPUSE	8	SEIN	4	2
C	48	CSE	12	COSTS	10	2
C	49	IH	8	SEIN	6	2
C	50	CSU	8	SEIN	7	2
C	51	MSE	8	SEIN	8	2
C	52	KTR	8	SEIN	9	2
C	53	ND	11	SCAL	67	1
C	54	UCDSE	9	DSE	2	2
C	55	COA	10	AIDATA	1	2

C	56	BRCT	11	SCAL	1	2
C	57	OS	11	SCAL	4	2
C	58	OSTO	11	SCAL	3	2
C	59	OSTC	11	SCAL	2	2
C	60	EBO	11	SCAL	5	2
C	61	KPSR	11	SCAL	6	2
C	62	SPRTS	11	SCAL	7	2
C	63	WRMC	11	SCAL	8	2
C	64	CNFL	11	SCAL	9	2
C	65	CTFL	11	SCAL	10	2
C	66	CTFX	11	SCAL	11	2
C	67	CNSL	11	SCAL	12	2
C	68	CNSS	11	SCAL	13	2
C	69	CTSL	11	SCAL	14	2
C	70	CTSS	11	SCAL	15	2
C	71	FJB	11	SCAL	16	2
C	72	NWK	6	AFIN	1	2
C	73	NMH	11	SCAL	19	2
C	74	NMMKW	11	SCAL	18	2
C	75	NCHMM	11	SCAL	20	2
C	76	CC	11	SCAL	26	2
C	77	COC	11	SCAL	22	2
C	78	CPH	12	COSTS	18	2
C	79	SNPC	11	SCAL	24	2
C	80	CUR	11	SCAL	25	2
C	81	CCPH	11	SCAL	21	2
C	82	SCC	11	SCAL	27	2
C	83	NSS	11	SCAL	28	2
C	84	SLR	11	SCAL	29	2
C	85	PC	11	SCAL	30	2
C	86	KTS	11	SCAL	31	2
C	87	KIH	8	SEIN	2	2
C	88	IMC	11	SCAL	33	2
C	89	RMC	11	SCAL	34	2
C	90	SA	11	SCAL	35	2
C	91	PSC	11	SCAL	36	2
C	92	PSO	11	SCAL	37	2
C	93	RPUM	11	SCAL	38	2
C	94	OHI	11	SCAL	39	2
C	95	COS	11	SCAL	40	2
C	96	ILR	6	AFIN	15	2
C	97	PTT	11	SCAL	42	2
C	98	CAC	12	COSTS	2	2
C	99	OSCY	11	SCAL	44	2
C	100	PMB	11	SCAL	45	2
C	101	EFF	11	SCAL	46	2
C	102	PIUP	11	SCAL	48	1
C	103	NACB	11	SCAL	49	2
C	104	CPA	11	SCAL	50	2
C	105	FHACH	11	SCAL	51	2
C	106	MFHACH	11	SCAL	52	2
C	107	FCL	4	RUIN	3	2

C	108	KSLPT	11	SCAL	54	3
C	109	KPJG	11	SCAL	55	2
C	110	KCJB	11	SCAL	56	2
C	111	AAOH	11	SCAL	57	2
C	112	BCA	11	SCAL	58	2
C	113	BPA	11	SCAL	59	2
C	114	FLA	11	SCAL	60	2
C	115	OBSEC	11	SCAL	61	2
C	116	CBTE	11	SCAL	62	2
C	117	CBCH	11	SCAL	63	2
C	118	CCIT	11	SCAL	64	2
C	119	IR	11	SCAL	65	2
C	120	KSED	11	SCAL	66	2
C	121	NDSER	9	DSE	1	1
C	122	CDSE	11	SCAL	68	2
C	123	CFB	11	SCAL	69	2
C	124	POT	11	SCAL	70	1
C	125	DPT	11	SCAL	71	1
C	126	DRCT	4	RUIN	6	2
C	127	CTFS	11	SCAL	73	2
C	128	CNFS	11	SCAL	74	2
C	129	CNFX	11	SCAL	75	2
C	130	CTSX	11	SCAL	76	2
C	131	CNSX	11	SCAL	77	2
C	132	YEAR	11	SCAL	78	1
C	133	COO	12	COSTS	1	2
C	134	CACQ	11	SCAL	43	2
C	135	COP	12	COSTS	3	2
C	136	CFL	12	COSTS	4	2
C	137	COM	12	COSTS	5	2
C	138	CSM	12	COSTS	6	2
C	139	CPT	12	COSTS	7	2
C	140	CSP	12	COSTS	8	2
C	141	CDR	12	COSTS	9	2
C	142	CSESH	8	SEIN	5	2
C	143	CSW	12	COSTS	11	2
C	144	CJB	12	COSTS	12	2
C	145	CIM	12	COSTS	13	2
C	146	CRD	12	COSTS	14	2
C	147	CSI	12	COSTS	15	2
C	148	COI	12	COSTS	16	2
C	149	CPP	12	COSTS	17	2
C	150	CPHM	11	SCAL	23	2
C	151	CPTI	12	COSTS	19	2
C	152	CSPI	12	COSTS	20	2
C	153	CDRI	12	COSTS	21	2
C	154	CSEI	12	COSTS	22	2
C	155	CSWI	12	COSTS	23	2
C	156	CJBI	12	COSTS	24	2
C	157	CIMI	12	COSTS	25	2
C	158	CFAI	12	COSTS	26	2
C	159	CDP	12	COSTS	27	2

\*\*\*\*\*

	AFIN(30,15)	RUIN(120,13)	SEIN(50,9)
	-----	-----	-----
1)	VN-1 - NWK	VI-1 - UC	VJ-1 - UCSE
2)	- ACB	- UCSRU	- KIH
3)	- CIC	- FCL	- KSE
4)	- COT	- FCS	- CPUSE
5)	- CTSB	- T	- CSESH
6)	- COJT	- DRCT	- IH
7)	- TCS	- DC	- CSU
8)	- TRS	- TC	- MSE
9)	VN-2 - CMPB	VI-2 - PS	VJ-2 - KTR
10)	- OPF	- PP	
11)	- KM	- SP	
12)	- DLR	- IC	
13)	- LLR	- CALI	
14)	- BMR		
15)	- ILR		

	SUBIN(40,4)	DSE(50,2)	AIDATA(50)
	-----	-----	-----
1)	VM-1 - CPINT	VD-1 - NDSER	VP-1 - COA
2)	- CINST	- UCDSE	
3)	- CFJG		
4)	- CSJG		

**00000000**

VE-1 - 1) COO	VE-2 - 10) CSE	VE-3 - 19) CPTI
- 2) CAC	- 11) CSW	- 20) CSPI
- 3) COP	- 12) CJB	- 21) CDRI
- 4) CFL	- 13) CIM	VE-4 - 22) CSEI
- 5) COM	VE-3 - 14) CRD	- 23) CSWI
- 6) CSM	- 15) CSI	- 24) CJB
- 7) CPT	- 16) COI	- 25) CIMI
- 8) CSP	- 17) CPP	- 26) CFAI
VE-2 - 9) CDR	- 18) CPM	- 27) CDP

.....

VS-1 - 1) BRCT	VS-4 - 27) SCC	VS-7 - 53) FC
- 2) OSTC	- 28) NSS	- 54) KSLPT
- 3) OSTO	- 29) SLR	- 55) KPJG
- 4) OS	- 30) PC	- 56) KCJG
- 5) EBO	- 31) KTS	VS-8 - 57) AAOH
- 6) KPSR	- 32) KI	- 58) BCA

```

C      - 7) SPRTS      VS-5 - 33) IMC      - 59) BPA
C      - 8) WRMC       - 34) RMC       - 60) FLA
C      VS-2 - 9) CNFL   - 35) SA       - 61) OBSEC
C      - 10) CTFL      - 36) PSC      - 62) CBTE
C      - 11) CTFX      - 37) PSD      - 63) CBCM
C      - 12) CNSL      - 38) RPUW      - 64) CCIT
C      - 13) CNSS      - 39) OHI      VS-9 - 65) IR
C      - 14) CTSL      - 40) COS      - 66) KSED
C      - 15) CTSS      VS-6 - 41) NOT USED - 67) ND
C      - 16) FJB       - 42) PTT      - 68) CDSE
C      VS-3 - 17) NW    - 43) CACQ      - 69) CFB
C      - 18) NMMKW     - 44) OSCY      - 70) POT
C      - 19) NMM       - 45) PMP      - 71) DPT
C      - 20) NCHMM     - 46) EFF      - 72) DR
C      - 21) CCPH      - 47) NB       VS-0 - 73) CTFS
C      - 22) COC       - 48) PIUP      - 74) CNFS
C      - 23) CPMH      VS-7 - 49) NACB      - 75) CNFX
C      - 24) SWPC      - 50) CPA       - 76) CTSX
C      VS-4 - 25) CUR   - 51) FHACH      - 77) CNSX
C      - 26) CC        - 52) MFHACH     - 78) YEAR
C
C      INTEGER      JWHERE(159),JWHICH(159),JFORM(159),LENGTH(10)
C      CHARACTER*1  IDS(10)
C      CHARACTER*10 PHEAD(2),PERT(3),VARY(161),NAME,BLANK
C      CHARACTER*17 FORM(4),FORMAT
C
C      COMMON /ALL/  NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
C      COMMON /LINES/ MAXLIN
C
C      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
C      CHARACTER*7  AFID(50),SEID(50)
C      CHARACTER*7  SEQID(40),LEQID(120)
C      COMMON /EQIDS/ SEQID,LEQID,AFID,SEID
C
C      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
C      *              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
C      COMMON /RAM/  SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
C      COMMON /COSTIO/ IOIN,IOUT,NTH
C
C      CHARACTER*1  XMASK,XTITLE
C      COMMON /JJF/  XMASK(10),XTITLE(10)
C
C      DATA JWHICH/11,1,1,4*2,5*3,4,4,11,4,4,11,7*4,5,5,11,13*6,
C      & 7,7,8,11,8,8,12,4*8,11,9,10,16*11,6,5*11,12,8*11,8,
C      & 8*11,6,11,12,8*11,4,13*11,9,4*11,4,6*11,12,11,7*12,8,7*12,
C      & 11,9*12/
C
C      DATA JWHERE/47,8,9,1,2,3,4,3,4,5,6,7,1,2,53,4,5,72,7,8,9,10,

```

```

* 11,12,13,1,2,17,2,3,4,5,6,7,8,9,10,11,12,13,14,1,2,1,32,3,
* 4,10,6,7,8,9,67,2,1,1,4,3,2,5,6,7,8,9,10,11,12,13,14,15,16,
* 1,19,18,20,26,22,18,24,25,21,27,28,29,30,31,2,33,34,35,36,
* 37,38,39,40,15,42,2,44,45,46,48,49,50,51,52,3,54,55,56,57,
* 58,59,60,61,62,63,64,65,66,1,68,69,70,71,6,73,74,75,76,77,
* 78,1,43,3,4,5,6,7,8,9,5,11,12,13,14,15,16,17,23,19,20,21,
* 22,23,24,25,26,27/

```

C

```

DATA JFORM/2,2,1,5*2,3*3,1,8*2,3*1,4*2,1,13*2,4,4,9*2,1,48*2,
* 1,5*2,3,12*2,1,2,2,1,1,6*2,1,27*2/

```

C

```

DATA LENGTH/4*7,5*5,7/
DATA BLANK/' '/
DATA FORM/('(1X,A7,1X,2F11.0)',
*          '(1X,A7,1X,2F11.2)',
*          '(1X,A7,1X,2F11.5)',
*          '(1X,A7,1X,2F11.6)'/
DATA PHEAD/'CHANGES','PERTURBED'/
DATA PERT/'BIAS','FACTOR','REPLACE'/

```

C

```

DATA VARY/'SET','GLOSSARY','NB','MFHBMA',
* 'SNSRU','CPINT','CINST','CFJB','CSJB','W','PM','PN',
* 'PS','LNSRU','UC','UCSRU','FC','FCS','T','DR','DC','TC',
* 'PA','PP','SP','IC','CALI','FMMH','SMMH','NW','ACB','CIC',
* 'COT','CTTS','COJT','TCS','TRS','CMPS','OPF','KH','DLR',
* 'LLR','BMR','TSDEN','TSDOT','UCSE','KI','KSE','CPUSE',
* 'CSE','IH','CSU','MSE','KTR','ND','UCDSE','COA','BRCT',
* 'OS','OSTO','OSTC','EBO','KPSR','SPRTS','WRMC','CNFL',
* 'CTFL','CTFX','CNSL','CNSS','CTSL','CTSS','FJB','NMK',
* 'NMH','NMHW','NCHMM','CC','COC','CPM','SWPC','CUR',
* 'CCPH','SCC','NSS','SLR','PC','KTS','KIH','INC','RMC',
* 'SA','PSC','PSO','RPW','OHI','COS','ILR','PTT','CAC',
* 'OSCY','PMB','EFF','PIUP','NACB','CPA','FHACN','MFHACN',
* 'FCL','KSLPT','KPJB','KCJB','AAOH','BCA','BPA','FLA',
* 'OBSEC','CBTE','CBCM','CCIT','IR','KSED','NDSE','CDSE',
* 'CFB','POT','DPT','DRCT','CTFS','CNFS','CNFX','CTSX',
* 'CNSX','YEAR','COO','CACQ','COP','CFL','COM','CSM','CPT',
* 'CSP','CDR','CSEH','CSW','CJB','CIM','CRD','CSI','COI',
* 'CPP','CPM','CPTI','CSPI','CDRI','CSEI','CSWI','CJBI',
* 'CIMI','CFAI','CDP'/

```

C

```

10 IF (NOTHER.EQ. 0) WRITE(*,11)
11 FORMAT(/,' COST VARIABLE?      ')
   CALL INP(4,VARY,0,0,161,160,J,*1000,*10)
   IF (J-2) 30,20,40
20 CALL DEFINE
   BOTO 10
30 CALL SET
   BOTO 10

```

C

C

```

GET PERTURBATION TYPE AND AMOUNT.

```

C

```

40 IF (NOTHER .EQ. 0) WRITE(*,41)
41 FORMAT(/,' TYPE?      ')
   CALL INP(4,PERT,0,0,3,100,KP,*10,*40)
50 IF (NOTHER .EQ. 0) WRITE(*,51) PERT(KP)
51 FORMAT(/,1X,A7,'=      ')
   CALL INP(2,0,0,0,0,110,VAL,*10,*50)

C
C SET POINTERS.  GET MASK AND PRINT OPTION.
C
   K=J-2
   JWK=JWHICH(K)
   L=JWHERE(K)
   FORMAT=FORM(JFORM(K))
   IF (JWK .GE. 9) GOTO 65
   IF (NMASK .EQ. -1) GOTO 60
   MK=NMASK
   GOTO 65
60 IF (NOTHER .EQ. 0) WRITE(*,61)
61 FORMAT(/,' MASK=      ')
   CALL INP(5,0,XMASK,0,LENGTH(JWK),120,MK,*10,*60)
65 IF (KLI .EQ. -1) GOTO 70
   IL=KLI
   GOTO 72
70 IF (NOTHER .EQ. 0) WRITE(*,71)
71 FORMAT(/,' DO YOU WANT A LISTING OF THE CHANGED ITEMS?      ')
   CALL INP(3,0,0,0,0,2,IL,*10,*70)

C
C BRANCH DEPENDING ON WHICH ARRAY DATA IS FOUND.
C
   72 KTOT=0
   NUMC=0
   GOTO (100,100,200,200,300,300,400,400,500,600,700,710),JWK
88 FORMAT(16X,'COST',4X,A10)

C
C IN THE NEXT 6 SECTIONS, THE CODE IS SIMILAR.  LABELS 100
C FOR SUBSYSTEMS, 200 FOR LRU'S, 300 FOR AFSC'S, 400 FOR SE'S,
C 500 FOR DSE'S, AND 600 FOR AIRCREW.  (THERE IS NO MASK FOR
C DSE AND AIRCREW AS THERE IS NO NAME FOR EACH.)
C
C FOR EACH TIME THROUGH THE LOOP, CHECK THE MASK (MK > 0),
C EXTRACT THE DATA PER JWK, CHECK FOR OTHER THAN REPLACE OF
C NONEXISTENT DATA, MODIFY WITH A CALL TO MODTYP, PRINT THE
C LINE, AND REPLACE THE NEW VALUE.
C
100 DO 150 I=1,NSUB
   NAME=SEQID(I)
   IF (MK .EQ. 0) GOTO 120
   READ(NAME,110) IDS
110 FORMAT(10A1)
   CALL MATCH(MK,XMASK,7,IDS,*150)
120 IF (JWK .EQ. 1) GOTO 130

```

```

      OLD=SIN(I,L)
      GOTO 135
130 OLD=SDAT(I,L,NTH)
135 IF (OLD .LE. -1.E30 .AND. KP .NE. 3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      NUMC=NUMC+1
      IF (IL .EQ. 0) GOTO 140
      CALL ABORT(KTOT,*140)
      IF (KTOT .EQ. 1) WRITE(*,88) PHEAD(IJ)
      WRITE(*,FORMAT) NAME,OLD,GNU
140 IF (JWK .EQ. 1) GOTO 145
      SIN(I,L)=GNU
      GOTO 150
145 SDAT(I,L,NTH)=GNU
150 CONTINUE
      GOTO 900

```

C

```

200 DO 250 I=1,NLRU
      NAME=LEQID(I)
      IF (MK .EQ. 0) GOTO 220
      READ(NAME,110) IDS
      CALL MATCH(MK,XMASK,7,IDS,*250)
220 IF (JWK .EQ. 3) GOTO 230
      OLD=RUIN(I,L)
      GOTO 235
230 OLD=UDAT(I,L,NTH)
235 IF (OLD .LE. -1.E30 .AND. KP .NE. 3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      NUMC=NUMC+1
      IF (IL .EQ. 0) GOTO 240
      CALL ABORT(KTOT,*240)
      IF (NUMC.EQ.1) WRITE(*,88) PHEAD(IJ)
      WRITE(*,FORMAT) NAME,OLD,GNU
240 IF (JWK .EQ. 3) GOTO 245
      RUIN(I,L)=GNU
      GOTO 250
245 UDAT(I,L,NTH)=GNU
250 CONTINUE
      GOTO 900

```

C

```

300 DO 350 I=1,NAF
      NAME=AFID(I)
      IF (MK .EQ. 0) GOTO 320
      READ(NAME,110) IDS
      CALL MATCH(MK,XMASK,5,IDS,*350)
320 IF (JWK .EQ. 5) GOTO 330
      OLD=AFIN(I,L)
      GOTO 335
330 OLD=ADAT(I,L,NTH)
335 IF (OLD .LE. -1.E30 .AND. KP .NE. 3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      NUMC=NUMC+1

```

```

      IF (IL.EQ.0) GOTO 340
      CALL ABORT(KTOT,*340)
      IF (KTOT.EQ.1) WRITE(*,88) PHEAD(IJ)
      WRITE(*,FORMAT) NAME,OLD,GNU
340  IF (JWK.EQ.5) GOTO 345
      AFIN(I,L)=GNU
      GOTO 350
345  ADAT(I,L,NTH)=GNU
350  CONTINUE
      GOTO 900

C
400  DO 450 I=1,NSE
      NAME=SEID(I)
      IF (MK.EQ.0) GOTO 420
      READ(NAME,110) IDS
      CALL MATCH(MK,XMASK,5,IDS,*450)
420  IF (JWK.EQ.7) GOTO 430
      OLD=SEIN(I,L)
      GOTO 435
430  OLD=EDAT(I,L,NTH)
435  IF (OLD.LE.-1.E30 .AND. KP.NE.3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      NUMC=NUMC+1
      IF (IL.EQ.0) GOTO 440
      CALL ABORT(KTOT,*440)
      IF (KTOT.EQ.1) WRITE(*,88) PHEAD(IJ)
      WRITE(*,FORMAT) NAME,OLD,GNU
440  IF (JWK.EQ.7) GOTO 445
      SEIN(I,L)=GNU
      GOTO 450
445  EDAT(I,L,NTH)=GNU
450  CONTINUE
      GOTO 900

C
500  IF (NDS.EQ.0) GOTO 800
      DO 550 I=1,NDS
      NAME=BLANK
      OLD=DSE(I,L)
      IF (OLD.LE.-1.E30 .AND. KP.NE.3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      NUMC=NUMC+1
      IF (IL.EQ.0) GOTO 540
      CALL ABORT(KTOT,*540)
      IF (KTOT.EQ.1) WRITE(*,88) PHEAD(IJ)
      WRITE(*,FORMAT) NAME,OLD,GNU
540  DSE(I,L)=GNU
550  CONTINUE
      GOTO 900

C
600  IF (NAI.EQ.0) GOTO 800
      DO 650 I=1,NAI
      NAME=BLANK

```

```

      OLD=AIDATA(I)
      IF (OLD .LE. -1.E30 .AND. KP .NE. 3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      NUMC=NUMC+1
      IF (IL .EQ. 0) GOTO 640
      CALL ABORT(KTOT,*640)
      IF (KTOT.EQ.1) WRITE(*,88) PHEAD(IJ)
      WRITE(*,FORMAT) NAME,OLD,GNU
640  AIDATA(I)=GNU
650  CONTINUE
      GOTO 900
C
C  SIMILARLY FOR SINGLE-VALUED VARIABLES FROM COSTS OR SCAL.
C
700  OLD=SCAL(L)
      GOTO 720
710  OLD=COSTS(L)
720  IF (OLD .LE. -1.E30 .AND. KP .NE. 3) GOTO 800
      CALL MODTYP(OLD,VAL,KP,GNU)
      IF (JWK .EQ. 12) GOTO 730
      SCAL(L)=GNU
      GOTO 740
730  COSTS(L)=GNU
740  IF (IL .NE. 1) GOTO 10
      WRITE(*,88) PHEAD(IJ)
      WRITE(NAME,745) VARY(J)
745  FORMAT(A7)
      WRITE(*,FORMAT) NAME,OLD,GNU
      GOTO 10
C
800  WRITE(*,801)
801  FORMAT(/,' NO DATA TO CHANGE.')
      NOTHER=0
      GOTO 10
C
C  IF R+M OUTPUT WAS CHANGED, RESET LAST TO FORCE RECOMPUTATION
C  IN RMODEL IF THE SAME INPUT FILE IS USED AGAIN.
C
900  WRITE(*,901) NUMC
901  FORMAT(//,1X,I4,' CHANGES.')
      IF (NUMC .EQ. 0) GOTO 10
      IF (JWK .EQ. 1 .OR. JWK .EQ. 3 .OR.
*      JWK .EQ. 5 .OR. JWK .EQ. 7) LAST=0
      GOTO 10
1000 RETURN
      END
      SUBROUTINE SCSI(CSI)
C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*      DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
      CALL SCPP(CPP,*100)

```

```

      CPM = COSTS(18)
      IF (CPM .LE. -1.E30) GO TO 100
      CSI = CPP + CPM
      CALL RITE(13,CSI)
      RETURN

C
100 CSI = COSTS(15)
   IF (CSI .LE. -1.E30) CALL ERROR('CSI  ')
   CALL RITE(13,CSI)
   RETURN
   END
   SUBROUTINE SCOI(COI)

C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)

C
      CJBI = COSTS(24)
      IF (CJBI .LE. -1.E30) GO TO 100
      CALL SCPTI(CPTI,*100)
      CALL SCSPI(CSPI,*100)
      CALL SCDRI(CDRI,*100)
      CALL SCSEI(CSEI,*100)
      CALL SCSWI(CSWI,*100)
      CALL SCIMI(CIMI,*100)
      CALL SCFAI(CFAI,*100)
      COI = CPTI + CSPI + CDRI + CSEI + CSWI + CJBI + CIMI + CFAI
      CALL RITE(12,COI)
      RETURN

C
100 COI = COSTS(16)
   IF (COI .LE. -1.E30) CALL ERROR('COI  ')
   CALL RITE(12,COI)
   RETURN
   END
   SUBROUTINE SCO(CO)
      CALL SCOP(COP,*100)
      CALL SCFL(CFL,*100)
      CO = COP + CFL
      CALL RITE(11,CO)
      RETURN

C
100 CALL ERROR('CO  ')
   RETURN
   END
   SUBROUTINE SCS(CS)
      CALL SCMSM(COM,CSM,*100)
      CALL SCPT(CPT,*100)
      CALL SCSP(CSP,*100)
      CALL SCDR(CDR,*100)
      CALL SCSE(CSE,*100)
      CALL SCSW(CSW,*100)
      CALL SCJB(CJB,*100)

```

```

CALL SCIM(CIM,*100)
CS = COM + CSM + CPT + CSP + CDR + CSE + CSW + CJB + CIM
CALL RITE(10,CS)
RETURN

C
100 CALL ERROR('CS      ')
RETURN
END
SUBROUTINE SCPTI(CPTI,*)
C
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
CBTE = SCAL(62)
IF (CBTE .LE. -1.E30) GO TO 10
CBCH = SCAL(63)
IF (CBCH .LE. -1.E30) GO TO 10
CCIT = SCAL(64)
IF (CCIT .LE. -1.E30) GO TO 10
CPTI = CBTE + CBCH + CCIT
CALL RITE(24,CPTI)
GO TO 20

C
10 CPTI = COSTS(19)
IF (CPTI .LE. -1.E30) RETURN 1
CALL RITE(24,CPTI)
20 RETURN
END
SUBROUTINE SCPP(CPP,*)
C
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
CPP = 0.0
DO 50 M=1,NSUB
I1 = KLRU(M)
I2 = I1 + NUML(M) - 1
CALL SCPINT(CPINT,I1,I2,*20)
GO TO 30
20 CPINT = SIN(M,1)
IF (CPINT .LE. -1.E30) RETURN 1
CALL RITE(36,CPINT)
30 CALL SCINST(CINST,I1,I2,*40)
GO TO 45
40 CINST = SIN(M,2)
IF (CINST .LE. -1.E30) RETURN 1
CALL RITE(37,CINST)
45 CPPS = EM * NACB * (CPINT + CINST)

```

```

      CALL RITE(63,CPPS)
      CPP = CPP + CPPS
50  CONTINUE
      CALL RITE(15,CPP)
      RETURN
      END
      SUBROUTINE SCSPI(CSPI,*)
C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
      COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
      COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
      COMMON /COSTIO/ IOIN,IOUT,NTH
C
      IF (EM .LE. -1.E30) GO TO 40
      JLRU=1
      M=1
      CSPI = 0.0
      DO 10 I=1,NLRU
      PS = UDAT(I,6,NTH)
      BRCT = SCAL(1)
      IF (BRCT .LE. -1.E30) GO TO 5
      PN = UDAT(I,5,NTH)
      OSTC = SCAL(2)
      IF (OSTC .LE. -1.E30) GO TO 5
      OS = SCAL(4)
      IF (OS .LE. -1.E30) GO TO 5
      OSTO = SCAL(3)
      IF (OSTO .LE. -1.E30) GO TO 5
      T = BRCT + PN/PS*(OSTC*(1.0-OS) + OSTO*OS - BRCT)
      RUIN(I,5) = T
      GO TO 7
5  T = RUIN(I,5)
      IF (T .LE. -1.E30) GO TO 40
C
7  CALL SLRUSS(ELRUSS,I,M,T,*40)
      CALL SLRUDS(ELRUDS,I,M,*40)
      CALL SSRUSS(SRUSS,I,M,T,*40)
      CALL SSRUDS(SRUDS,I,M,*40)
      JLRU=JLRU+1
      IF (JLRU .LE. NUML(M)) GO TO 8
      JLRU=1
      M=M+1
8  CSPIL = EM + (ELRUSS + ELRUDS + SRUSS + SRUDS)
      CALL RITE(63,CSPIL)
      CSPI = CSPI + CSPIL
10 CONTINUE

```

```

C      CALL SSPRTS(SPRTS,*20)
      GO TO 30
20  SPRTS = SCAL(7)*EM
      IF (SPRTS .LE. -1.E30) GO TO 40
      CALL RITE(49,SPRTS)
30  WRMC = SCAL(8)
      IF (WRMC .LE. -1.E30) GO TO 40
      CSPI = CSPI + SPRTS + WRMC
      CALL RITE(29,CSPI)
      GO TO 50

C
40  CSPI = COSTS(20)
      IF (CSPI .LE. -1.E30) RETURN 1
      CALL RITE(29,CSPI)
50  RETURN
      END
      SUBROUTINE SCDRI(CDRI,*)

C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)

C
      END = SCAL(67)
      IF (END .LE. -1.E30) GO TO 10
      CALL SCDSE(CDSE,*10)
      CDRI = END * CDSE
      CALL RITE(23,CDRI)
      GO TO 20

C
10  CDRI = COSTS(21)
      IF (CDRI .LE. -1.E30) RETURN 1
      CALL RITE(23,CDRI)
20  RETURN
      END
      SUBROUTINE SCSEI(CSEI,*)

C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)

C
      COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII

C
      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI

C
      TEMP = 0.0
      DO 70 J=1,NSE
      CPUSE = SEIN(J,4)
      IF (CPUSE .LE. -1.E30) GO TO 100
      CSESH = SEIN(J,5)
      IF (CSESH .LE. -1.E30) GO TO 100
      EYEH = SEIN(J,6)
      IF (EYEH .LE. -1.E30) GO TO 100
60  CSU = SEIN(J,7)

```

```

      IF (CSU .LE. -1.E30) GO TO 100
      TEMP = TEMP + EM*(CPUSE+CSESH+EYEH) + CSU
70  CONTINUE
      BCA = SCAL(58)
      IF (BCA .LE. -1.E30) GO TO 80
      BPA = SCAL(59)
      IF (BPA .LE. -1.E30) GO TO 80
      FLA = SCAL(60)
      IF (FLA .LE. -1.E30) GO TO 80
      OBSEC = BCA + BPA + FLA
      GO TO 90
80  OBSEC = SCAL(61)
90  IF (OBSEC .LE. -1.E30) GO TO 100
      CSEI = TEMP + EM * OBSEC
      CALL RITE(30,CSEI)
      GO TO 110
C
100 CSEI = COSTS(22)
      IF (CSEI .LE. -1.E30) RETURN 1
      CALL RITE(30,CSEI)
110 RETURN
      END
      SUBROUTINE SCSWI (CSWI,*)
C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *              DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
      ENW = SCAL(17)
      IF (ENW .LE. -1.E30) GO TO 10
      ENMMKW = SCAL(18)
      IF (ENMMKW .LE. -1.E30) GO TO 10
      ENMM = ENMMKW * ENW * .001
      GO TO 20
C
10  ENMM = SCAL(19)
      IF (ENMM .LE. -1.E30) GO TO 30
C
20  ENCHMM = SCAL(20)
      IF (ENCHMM .LE. -1.E30) GO TO 30
      CCPH = SCAL(21)
      IF (CCPH .LE. -1.E30) GO TO 30
      CPMH = SCAL(23)
      IF (CPMH .LE. -1.E30) GO TO 30
C
      COC = ENCHMM * CCPH * ENMM
      SWPC = ENMM * CPMH
      GO TO 40
C
30  SWPC = SCAL(24)
      IF (SWPC .LE. -1.E30) GO TO 50
      COC = SCAL(22)
      IF (COC .LE. -1.E30) GO TO 50

```

```

40 CSWI = SWPC + COC
   CALL RITE(31,CSWI)
   GO TO 60
C
50 CSWI = COSTS(23)
   IF (CSWI .LE. -1.E30) RETURN 1
   CALL RITE(31,CSWI)
60 RETURN
   END
   SUBROUTINE SCJBI
C
   COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
   COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUHL(40),NAF,NSE,NDS,NAI
C
   CJBI = 0.0
   DO 40 M=1,NSUB
   FJB = SCAL(16)
   IF (FJB .LE. -1.E30) GO TO 50
   CALL SCFJB(CFJB,M,*50)
   CALL SCBJB(CSJB,M,*50)
   CJBIS = (1 + FJB) * (CFJB + CSJB)
   CALL RITE(62,CJBIS)
   CJBI = CJBI + CJBIS
40 CONTINUE
   COSTS(24) = CJBI
   CALL RITE(26,CJBI)
   RETURN
50 CALL RITE(26,COSTS(24))
   RETURN
   END
   SUBROUTINE SCIMI(CIMI,*)
C
   COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
   COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
   COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUHL(40),NAF,NSE,NDS,NAI
C
   EYMC = SCAL(33)
   IF (EYMC .LE. -1.E30) GO TO 20
C
   CIMI = 0.0
   DO 10 I=1,NLRU
   PA = RUIN(I,9)
   PP = RUIN(I,10)
   IF (PA .LE. -1.E30 .OR. PP .LE. -1.E30) GO TO 20
   CIMIL = EYMC * (1 + PA + PP)
   CALL RITE(68,CIMIL)
   CIMI = CIMI + CIMIL
10 CONTINUE
   CALL RITE(34,CIMI)

```

```

RETURN
20 CIMI = COSTS(25)
  IF (CIMI .LE. -1.E30) RETURN 1
  CALL RITE(34,CIMI)
  RETURN
  END
  SUBROUTINE SCOP(COP,*)
C
  COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
  *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
  COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
  COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
  DATA X/-1.E30/
C
  COO = COSTS(1)
  IF (COO .LE. -1.E30) GO TO 10
  OSCY = SCAL(44)
  CPA = SCAL(50)
  IF (CPA .LE. X .OR. EM .LE. X .OR.
  *   COO .LE. X .OR. OSCY .LE. X) GO TO 10
  TEMP = 0.0
  DO 5 IP = 1,NAI
  COA = AIDATA(IP)
  IF (COA .LE. X) GO TO 10
  TEMP = TEMP + COA
  5 CONTINUE
  CAC = EM * NACB * CPA * (TEMP + OSCY*NAI)
  COP = CAC + COO
  CALL RITE(59,CAC)
  CALL RITE(22,COP)
  GO TO 20
10 COP = COSTS(3)
  IF (COP .LE. -1.E30) RETURN 1
  CALL RITE(22,COP)
20 RETURN
  END
  SUBROUTINE SCFL(CFL,*)
C
  COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
  *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
  COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
  FC = SCAL(53)
  IF (FC .LE. -1.E30) GO TO 10
  CFL = EM*ABFH*FC
  CALL RITE(23,CFL)
  GO TO 20
10 CFL = COSTS(4)
  IF (CFL .LE. -1.E30) RETURN 1

```

```

      CALL RITE(23,CFL)
20  RETURN
      END
      SUBROUTINE SCPT(CPT,*)
C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
      COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
      COMMON /COSTIO/ IOIN,IOUT,NTH
C
      COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
      EFF = SCAL(46)
      PMB = SCAL(45)
      PIUP = SCAL(48)
      IF (EFF .LE. -1.E30 .OR. PMB .LE. -1.E30 .OR.
      *   PIUP .LE. -1.E30) GO TO 40
      CPT = 0.0
      DO 30 N=1,NAF
      CALL STCS(TCS,N,*40)
      EMURF = ADAT(N,1,NTH)
      EMURS = ADAT(N,2,NTH)
      IF (EMURF .LE. -1.E30 .OR. EMURS .LE. -1.E30) GO TO 40
      EMU = (EMURF+EMURS) * ABFH/(EFF+PMB)
      TRS = AFIN(N,8)
      IF (TRS .LE. -1.E30) GO TO 40
      CPT = CPT + (1.0/PIUP+TRS) * EMU * TCS
30  CONTINUE
      CPT = CPT * EM
      CALL RITE(14,CPT)
      GO TO 50
40  CPT = COSTS(7)
      IF (CPT .LE. -1.E30) RETURN 1
      CALL RITE(14,CPT)
50  RETURN
      END
      SUBROUTINE SCDR(CDR,*)
C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
      COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
      COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
      COMMON /COSTIO/ IOIN,IOUT,NTH

```

C

```

M=1
JLRU=1
CDR = 0.0
DO 20 I=1,NLRU
FHBMA = SDAT(M,8,NTH)
JLRU = JLRU + 1
IF (JLRU .LE. NUML(M)) GO TO 5
JLRU=1
M=M+1
5 PN = UDAT(I,5,NTH)
DC = RUIN(I,7)
IF (DC .LE. -1.E30) GO TO 30
CALL STC(TC,I,*10)
GO TO 15
10 TC = RUIN(I,8)
IF (TC .LE. -1.E30) GO TO 30
15 CDRL = EM * ABFH *PN * (DC+TC)/FHBMA
CALL RITE(60,CDRL)
CDR = CDR + CDRL
20 CONTINUE
COS = SCAL(40)
IF (COS .LE. -1.E30) GO TO 30
OHI = SCAL(39)
IF (OHI .LE. -1.E30) GO TO 30
CDR = CDR + EM * NACB * COS * OHI
CALL RITE(17,CDR)
GO TO 40
30 CDR = COSTS(9)
IF (CDR .LE. -1.E30) RETURN 1
CALL RITE(17,CDR)
40 RETURN
END
SUBROUTINE SCJB(CJB,*)

```

C

```

COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)

```

C

```

EKPJB = SCAL(55)
IF (EKPJB .LE. -1.E30) GO TO 10
EKCJB = SCAL(56)
IF (EKCJB .LE. -1.E30) GO TO 10
CJB1 = COSTS(24)
IF (CJB1 .LE. -1.E-30) GO TO 10
CJB = EKPJB * EKCJB * CJB1
CALL RITE(20,CJB)
GO TO 20
10 CJB = COSTS(12)
IF (CJB .LE. -1.E30) RETURN 1
CALL RITE(20,CJB)
20 RETURN
END

```

```

SUBROUTINE SCMSM (COM,CSM,*)
C
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*          DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
COMMON /COSTIO/ IOIN,IOUT,NTH
C
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
DATA X/-1.E30/
C
EFF = SCAL(46)
IF (ABFH .LE. X .OR. EFF .LE. X .OR. EFF .EQ. 0.0) GO TO 45
CSM = 0.0
COM = 0.0
C
DO 40 N=1,NAF
CALL SLLR(ELLR,N,*45)
BMR = AFIN(N,14)
IF (BMR .LE. X) GO TO 45
PROD = ELLR*BMR
EMURF = ADAT(N,1,NTH)
COM = COM + EMURF*PROD
EMURS = ADAT(N,2,NTH)
CSM = CSM + EMURS*PROD
40 CONTINUE
C
COM = EM * COM * ABFH/EFF
CSM = EM * CSM * ABFH/EFF
CALL RITE(27,COM)
CALL RITE(28,CSM)
RETURN
C
45 COM = COSTS(5)
IF (COM .LE. -1.E30) RETURN 1
CSM = COSTS(6)
IF (CSM .LE. -1.E30) RETURN 1
CALL RITE(27,COM)
CALL RITE(28,CSM)
RETURN
END
SUBROUTINE SCSP (CSP,*)
C
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*          DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
COMMON /COSTIO/ IOIN,IOUT,NTH
C

```

```

COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
C
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
CSP = 0.0
JLRU=1
M=1
IF (ABFH .LE. -1.E30) GO TO 20
DO 10 I=1,NLRU
FHBMA = SDAT(M,8,NTH)
PN = UDAT(I,5,NTH)
PW = UDAT(I,4,NTH)
UC = RUIN(I,1)
IF (UC .LE. -1.E30) GO TO 20
FCL = RUIN(I,3)
IF (FCL .LE. -1.E30) GO TO 20
UCSRU = RUIN(I,2)
IF (UCSRU .LE. -1.E30) CALL SUCSRU(UCSRU,I,*20)
FCS = RUIN(I,4)
IF (FCS .LE. -1.E30) GO TO 20
JLRU=JLRU+1
IF (JLRU.LE.NUML(M)) GO TO 5
JLRU=1
M=M+1
5 ELRURS = EM * ABFH * UC * FCL * PN / FHBMA
SRURS = EM * ABFH * UCSRU * FCS * PW / FHBMA
CSPL = ELRURS + SRURS
CALL RITE(70,ELRURS)
CALL RITE(71,SRURS)
CALL RITE(61,CSPL)
CSP = CSP + CSPL
10 CONTINUE
CALL RITE(16,CSP)
GO TO 30
20 CSP = COSTS(8)
IF (CSP .LE. -1.E30) RETURN 1
CALL RITE(16,CSP)
30 RETURN
END
SUBROUTINE SCSE(CSE,*)
C
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
CSE = 0.0
DO 20 J=1,NSE

```

```

      EMSE = SEIN(J,8)
      IF (EMSE .LE. -1.E30) GO TO 30
      CPUSE = SEIN(J,4)
      IF (CPUSE .LE. -1.E30) GO TO 30
      CSE = CSE + EMSE*CPUSE
20  CONTINUE
      CSE = EM * CSE
      CALL RITE(18,CSE)
      GO TO 40
30  CSE = COSTS(10)
      IF (CSE .LE. -1.E30) RETURN 1
      CALL RITE(18,CSE)
40  RETURN
      END
      SUBROUTINE SCSW (CSW,*)
C
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
      ENSS = SCAL(28)
      IF (ENSS .LE. -1.E30) GO TO 20
      SLR = SCAL(29)
      IF (SLR .LE. -1.E30) GO TO 10
      PC = ENSS * SLR
      GO TO 15
10  PC = SCAL(30)
      IF (PC .LE. -1.E30) GO TO 20
15  CUR = SCAL(25)
      IF (CUR .LE. -1.E30) GO TO 20
      CC = SCAL(26)
      IF (CC .LE. -1.E30) GO TO 20
      SCC = CUR * CC * ENSS * 12.0
      CSW = PC + SCC
      CALL RITE(19,CSW)
      GO TO 30
20  CSW = COSTS(11)
      IF (CSW .LE. -1.E30) RETURN 1
      CALL RITE(19,CSW)
30  RETURN
      END
      SUBROUTINE SLRUSS (ELRUSS,I,M,T,*)
      COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
      *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
      COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
      COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
      COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
      COMMON /COSTIO/ IOIN,IOUT,NTH
      DIMENSION UC(120)
      EQUIVALENCE (UC(1),RUIN(1,1))
C
      FHBMA = SDAT(M,8,NTH)
      PS = UDAT(I,6,NTH)

```

```

ELAMI = PBFH * PS / (FHBMA*12.0)
IF (T .LE. -1.E30) RETURN 1
CALL SKSTK(ELAMI,T,KSTK)
ELRUSS = KSTK * UC(1)
CALL RITE(33,FLOAT(KSTK))
CALL RITE(32,ELRUSS)
RETURN
END
SUBROUTINE SKSTK(ELAM,T,KSTK)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),BEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
EQUIVALENCE (EBO,SCAL(5))
TL = T * ELAM
KSTK = 0
IF (TL .LE. EBO) RETURN
IF (TL .LT. 80.) GO TO 5
KSTK=1
RETURN
5 L=0
SUM=0.0
EXPMT=EXP(-TL)
10 KSTK=KSTK+1
SUM=SUM+(TL**L)*EXPMT/FACT(L)
ANS=(TL-KSTK)*SUM+KSTK*(TL**KSTK)*EXPMT/FACT(KSTK)
L=L+1
IF (ANS .GT. EBO) GO TO 10
RETURN
END
FUNCTION FACT(N)
DIMENSION F(32),B(20)
DATA F/1.,1.,2.,6.,24.,120.,720.,5040.,40230.,362880.,3628800.,
& 3991680.,47900160.,6.2270208E9,8.7178291E10,1.3076744E12,
& 2.092790E13,3.5568743E14,6.4023737E15,1.2164510E17,
& 2.4329020E18,5.1090942E19,1.1240007E21,2.8582017E22,
& 6.2044840E23,1.5511210E25,4.0329146E26,1.0888869E28,
& 3.0488834E29,8.8417620E30,2.6525286E32,8.2228387E33/
B(2)=.16666666
B(4)=-.03333333
B(6)=.02380953
B(8)=-.03333333
B(10)=.07575757
B(12)=-.23113553
B(14)=1.16666666
B(16)=-7.09216686
B(18)=54.97117794
B(20)=-529.12424242
M = N+1
IF (M .GT. 32) GO TO 10
FACT = F(M)
RETURN
10 SUM=0.0
AM=M

```

```

DO 20 J=1,10
  R=J*2.0
  SUM=B(J*2)/(R*(R-1.0)*(AM** (R-1.0)))
20 CONTINUE
ALNX=(AM-.5)*ALOG(AM)-AM+0.91893853206+SUM
FACT=EXP(ALNX)
RETURN
END
SUBROUTINE SCPINT (CPINT,I1,I2,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
DIMENSION UC(120)
EQUIVALENCE (UC(1),RUIN(1,1))

C
CPINT = 0.0
AKTS = SCAL(31)
DO 20 I=I1,I2
  UCI=UC(I)
  IF (UCI .LE. -1.E30) RETURN 1
  AIC = RUIN(I,12)
  IF (AIC .GT. -1.E30) GO TO 18
  IF (AKTS .LE. -1.E30) RETURN 1
  AIC = AKTS * UCI
  RUIN(I,12) = AIC
18 CPINTL = UCI + AIC
  CALL RITE(66,CPINTL)
  CPINT = CPINT + CPINTL
20 CONTINUE
  CALL RITE(36,CPINT)
  RETURN
  END
SUBROUTINE SCINST(CINST,I1,I2,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
DIMENSION UC(120)
EQUIVALENCE(UC(1),RUIN(1,1))

C
CINST = 0.0
AKI = SCAL(32)
DO 20 I=I1,I2
  CALI = RUIN(I,13)
  IF (CALI .GT. -1.E30) GO TO 18
  UCI = UC(I)
  IF (UCI .LE. -1.E30) RETURN 1
  IF (AKI .LE. -1.E30) RETURN 1
  CALI = AKI * UCI
18 CINSTL = CALI
  CALL RITE(67,CINSTL)
  CINST = CINST + CINSTL
20 CONTINUE

```

```

CALL RITE(37,CINST)
RETURN
END
SUBROUTINE SLRUDS (ELRUDS,I,M,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
COMMON /COSTID/ IOIN,IOUT,NTH
COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
DIMENSION UC(120)
EQUIVALENCE (UC(1),RUIN(1,1))

```

C

```

IF (PBFH .LE. -1.E30) RETURN 1
UCI = UC(I)
IF (UCI .LE. -1.E30) RETURN 1
PN = UDAT(I,5,NTH)
FHBMA = SDAT(M,8,NTH)
DRCT = RUIN(I,6)
IF (DRCT .LE. -1.E30) RETURN 1
DPLL = PBFH * PN * DRCT / FHBMA
ELRUDS = DPLL * UCI
CALL RITE(69,DPLL)
CALL RITE(38,ELRUDS)
RETURN
END

```

```

SUBROUTINE STC (TC,I,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
COMMON /COSTID/ IOIN,IOUT,NTH
DATA X/-1.E30/

```

C

```

RPUW = SCAL(38)
PSC = SCAL(36)
OS = SCAL(4)
PSO = SCAL(37)
IF (RPUW .LE. X .OR. PSC .LE. X .OR.
* PSO .LE. X .OR. OS .LE. X) GO TO 10
W = UDAT(I,3,NTH)
TC = W * RPUW * 2.0 * (PSC * (1.0-OS) + PSO * OS)
CALL RITE(39,TC)
RETURN

```

C

```

10 TC = RUIN(I,8)
IF (TC .LE. X) RETURN 1
CALL RITE(39,TC)
RETURN
END
SUBROUTINE STCS (TCS,N,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
DATA X/-1.E30/

```

C

```

COJT = AFIN(N,6)
IF (COJT .LE. X) RETURN 1
ENWK = AFIN(N,1)
ACB = AFIN(N,2)
CIC = AFIN(N,3)
COT = AFIN(N,4)
PTT = SCAL(42)
CACB = SCAL(43)
IF (ENWK .LE. X .OR. ACB .LE. X .OR.
*   CIC .LE. X .OR. COT .LE. X .OR.
*   COT .LE. X .OR. PTT .LE. X) GO TO 10
CTTS = ENWK * (ACB+CIC) + PTT + COT + CACB
AFIN(N,5) = CTTS
GO TO 20

```

C

```

10 CTTS = AFIN(N,5)
IF (CTTS .LE. X) RETURN 1
20 TCS = CTTS + COJT
CALL RITE(40,TCS)
RETURN
END
SUBROUTINE SSRUDS(SRUDS,I,M,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*   DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
COMMON /COSTIO/ IOIN,IOUT,NTH

```

C

```

PW = UDAT(I,4,NTH)
FHBMA = SDAT(M,8,NTH)
RCT = RUIN(I,6)
IF (DRCT .LE. -1.E30 .OR. PBFH .LE. -1.E30) GO TO 10
DPLS = PBFH * PW * DRCT / FHBMA
UCSRU = RUIN(I,2)
IF (UCSRU .LE. -1.E30) CALL SUCSRU(UCSRU,I,*10)
SRUDS = DPLS * UCSRU
CALL RITE(73,DPLS)
CALL RITE(41,SRUDS)
RETURN
10 RETURN 1
END
SUBROUTINE SCFJB (CFJB,M,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*   DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
COMMON /COSTIO/ IOIN,IOUT,NTH
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
DATA X/-1.E30/
NSRU = SDAT(M,9,NTH)
CNFL = SCAL(9)
CTFL = SCAL(10)

```

```

CTFS = SCAL(73)
CNFS = SCAL(74)
CTFX = SCAL(11)
CNFX = SCAL(75)
IF (CNFL .LE. X .OR. CTFL .LE. X .OR.
*   CTFS .LE. X .OR. CNFS .LE. X .OR.
*   CTFX .LE. X .OR. CNFX .LE. X) GO TO 10
CFJB = NUML(M) * (CNFL+CTFL) + NSRU * (CTFS+CNFS) + CTFX + CNFX
CALL RITE(42,CFJB)
RETURN

```

C

```

10 CFJB = SIN(M,3)
IF (CFJB .LE. X) RETURN 1
CALL RITE(42,CFJB)
RETURN
END
SUBROUTINE SCSJB (CSJB,M,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*   DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /RAM/ SDAT(40,9,2),UDAT(120,7,2),ADAT(50,3,2),EDAT(50,2,2)
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
COMMON /COSTIO/ IOIN,IOUT,NTH
DATA X/-1.E30/

```

C

```

CNSL = SCAL(12)
CTSL = SCAL(14)
CTSS = SCAL(15)
CNSS = SCAL(13)
CTSX = SCAL(76)
CNSX = SCAL(77)
IF (CNSL .LE. X .OR. CTSL .LE. X .OR.
*   CTSS .LE. X .OR. CNSS .LE. X .OR.
*   CTSX .LE. X .OR. CNSX .LE. X) GO TO 10
NSRU = SDAT(M,9,NTH)
CSJB = NUML(M) * (CNSL+CTSL) + NSRU * (CTSS+CNSS) + CTSX + CNSX
CALL RITE(43,CSJB)
RETURN

```

C

```

10 CSJB = SIN(M,4)
IF (CSJB .LE. X) RETURN 1
CALL RITE(43,CSJB)
RETURN
END
SUBROUTINE SCPUSE
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*   DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
DIMENSION CPUSE(50),CSERM(50),EYEH(50)
EQUIVALENCE (CPUSE(1),SEIN(1,4)),(CSERM(1),SEIN(1,5))
EQUIVALENCE (EYEH(1),SEIN(1,6))

```

C

```

DO 10 J=1,NSE

```

```

CALL RITE(57,FLOAT(KSTK))
CALL RITE(47,SRUSS)
RETURN
10 RETURN 1
END
SUBROUTINE SLLR (ELLR,N,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
DATA X/-1.E30/
C
EYELR = AFIN(N,15)
OSCY = SCAL(44)
PMB = SCAL(45)
IF (EYELR .LE. X .OR. OSCY .LE. X .OR.
* PMB .LE. X .OR. PMB .EQ. 0.0) RETURN 1
AKM = AFIN(N,11)
CMPS = AFIN(N,9)
OPF = AFIN(N,10)
IF (AKM .LE. X .OR. CMPS .LE. X .OR. OPF .LE. X) GO TO 10
DLR = AKM * (CMPS + OPF)
AFIN(N,12) = DLR
GO TO 20
C
10 DLR = AFIN(N,12)
IF (DLR .LE. X) RETURN 1
20 ELLR = DLR + EYELR + OSCY /PMB
CALL RITE(48,ELLR)
RETURN
END
SUBROUTINE SSPTS (SPRTS,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
AKSPR = SCAL(6)
IF (AKSPR .LE. -1.E30) RETURN 1
UC = 0.0
DO 10 I=1,NLRU
UC = UC + RUIN(I,1)
10 CONTINUE
SPRTS = AKSPR * UC * EM
CALL RITE(49,SPRTS)
RETURN
END
SUBROUTINE SCDSE (CDSE,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
CDSE = 0.0
AKSED = SCAL(66)

```

```

COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
* DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
COMMON /COSTIO/ IOIN,IOUT,NTH
C
CHARACTER*7 AFID,SEID
CHARACTER*7 SEQID,LEQID
COMMON /EQIDS/ SEQID(40),LEQID(120),AFID(50),SEID(50)
C
WRITE(IOUT) 0, 0.0
WRITE(IOUT-9,999) 0, 0.0
999 FORMAT(I10,2X,E15.8)
DO 10 N=1,NSUB
WRITE(IOUT) SEQID(N),(SDAT(N,I,1),I=1,9),(SIN(N,J),J=1,4)
WRITE(IOUT-9,998) SEQID(N),(SDAT(N,I,1),I=1,9),(SIN(N,J),J=1,4)
998 FORMAT(A7,/,7(E15.8,2X),/,6(E15.8,2X))
10 CONTINUE
C
DO 30 N=1,NLRU
WRITE(IOUT) LEQID(N),(UDAT(N,J,1),J=2,7),(RUIN(N,J),J=1,13)
WRITE(IOUT-9,997) LEQID(N),(UDAT(N,J,1),J=2,7),(RUIN(N,J),J=1,13)
997 FORMAT(A7,/,7(E15.8,2X),/,7(E15.8,2X),/,5(E15.8,2X))
30 CONTINUE
C
DO 50 N=1,NAF
WRITE(IOUT) AFID(N),ADAT(N,1,1),ADAT(N,2,1),(AFIN(N,J),J=1,15)
WRITE(IOUT-9,996) AFID(N),ADAT(N,1,1),ADAT(N,2,1),
* (AFIN(N,J),J=1,15)
996 FORMAT(A7,/,7(E15.8,2X),/,7(E15.8,2X),/,3(E15.8,2X))
50 CONTINUE
C
DO 70 N=1,NSE
WRITE(IOUT) SEID(N),EDAT(N,1,1),EDAT(N,2,1),(SEIN(N,J),J=1,9)
WRITE(IOUT-9,995) SEID(N),EDAT(N,1,1),EDAT(N,2,1),
* (SEIN(N,J),J=1,9)
995 FORMAT(A7,/,7(E15.8,2X),/,4(E15.8,2X))
70 CONTINUE
IF (NAI.EQ.0) GO TO 110
DO 90 N=1,NAI
WRITE(IOUT) N,AIDATA(N)
WRITE(IOUT-9,994) N,AIDATA(N)
994 FORMAT(I10,2X,E15.8)
90 CONTINUE
C
110 WRITE(IOUT) COSTS
WRITE(IOUT-9,993) COSTS
993 FORMAT(12(7(E15.8,2X),/),)
WRITE(IOUT) SCAL

```

```

WRITE(IOUT-9,993) SCAL
C
WRITE(IOUT)      EM,ABFH,FLOAT(NACB),PBFH,ENNII
WRITE(IOUT-9,992) EM,ABFH,FLOAT(NACB),PBFH,ENNII
992 FORMAT(5(E15.8,2X))
C
WRITE(IOUT)      NUML
WRITE(IOUT-9,991) NUML
991 FORMAT(4(10I12,/))
RETURN
END
SUBROUTINE SCIM (CIM,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*             DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
COMMON /SIZES/ NSUB,NLRU,KLRU(40),NUML(40),NAF,NSE,NDS,NAI
C
CIM = 0.0
RMC = SCAL(34)
IF (RMC .LE. -1.E30) GO TO 20
SA = SCAL(35)
IF (SA .LE. -1.E30) GO TO 20
DO 10 I=1,NLRU
SP = RUIN(I,11)
PP = RUIN(I,10)
PA = RUIN(I,9)
IF (SP .LE. -1.E30 .OR. PP .LE. -1.E30 .OR.
*   PA .LE. -1.E30) GO TO 20
ENNII = 1 + PA + PP
BLII = ENNII + SP
CIML = RMC + ENNII + EM*SA*BLII
CALL RITE(64,CIML)
CIM = CIM + CIML
10 CONTINUE
CALL RITE(21,CIM)
RETURN
C
20 CIM = COSTS(13)
IF (CIM .LE. -1.E30) RETURN 1
CALL RITE(21,CIM)
RETURN
END
SUBROUTINE SCFAI(CFAI,*)
COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
*             DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
COMMON /BASIC/ EM,ABFH,NACB,PBFH,ENNII
C
CFB = SCAL(69)
IF (CFB .LE. -1.E30) GO TO 10
CFAI = EM + CFB
CALL RITE(35,CFAI)
GO TO 20

```

```

C
10 CFAI = COSTS(26)
   IF (CFAI .LE. -1.E30) RETURN 1
   CALL RITE(35,CFAI)
20 RETURN
   END
   SUBROUTINE XOUT

C
C THIS ROUTINE IS CALLED BY MAIN TO DISPLAY OUTPUTS FROM R+M
C MODEL (/RAM/) OR COST MODEL (OUTPUT FILES IO3A, IO4A).
C
C NMAX = 16 - ONLY R+M OUTPUTS AVAILABLE
C       66 - ALL OUTPUTS AVAILABLE
C   IO1 = R+M BASE FILE
C   IO2 = OPTIONAL R+M PERTURBED FILE
C   IO3A = BASE OUTPUT FILE
C   IO4A = OPTIONAL PERTURBED OUTPUT FILE
C   CONDITIONAL RETURN - TERMINATE THE PROGRAM
C
C *****
C
C                               /RAM/
C
C          SDAT          UDAT          ADAT          EDAT
C          ----          ----          ----          ----
C 1        MTTRS          MTTR          * FMMH          * TSDEM
C 2        MTTRF          MMH           * SMMH          * TSDDOT
C 3        MTTRT          * WEIGHT      MMHT
C 4        MMHS           * PW
C 5        MMHF           * PN
C 6        MMHT           * PS
C 7        AVAIL          * LNSRU
C 8        * MFH8MA
C 9        * SNSRU
C
C * - DENOTES COST MODEL INPUT
C
C
C   REAL STORA(120),STORB(120)
C   INTEGER JTOT(49),JADD(13)
C   INTEGER DUMA(20),DUMN(6)
C   INTEGER JTYP(49),KOUNT(5)
C   CHARACTER*10 OUTS(66)
C   CHARACTER*7  BLANK,TEMP(30)
C
C   COMMON /OVER/ JABT,IO2,IO4A,JN,NMAX,NLAST
C
C   INTEGER MASK,TITLE
C   COMMON /ALL/ NOTHER,KPR,KSD,KLI,NMASK,NTITL,MASK(10),TITLE(10)

```



C	19	CO	1	1	1
C	20	COI	1	1	1
C	21	CSI	1	1	1
C	22	CPT	1	1	1
C	23	CPP	1	1	1
C	24	CSP	1	1	1
C	25	CDR	1	1	1
C	26	CSE	1	1	1
C	27	CSW	1	1	1
C	28	CJ8	1	1	1
C	29	CIN	1	1	1
C	30	COP	1	1	1
C	31	CFL	1	1	1
C	32	CPTI	1	1	1
C	33	CDRI	1	1	1
C	34	CJ8I	1	1	1
C	35	COM	1	1	1
C	36	CSM	1	1	1
C	37	CSPI	1	1	1
C	38	CSEI	1	1	1
C	39	CSWI	1	1	1
C	40	LRUSS	1	NLRU	3
C	41	STKL	0	NLRU	3
C	42	CIMI	1	1	1
C	43	CFAI	1	1	1
C	44	CPINT	1	NSUB	2
C	45	CINST	1	NSUB	2
C	46	LRUDS	1	NLRU	3
C	47	TC	1	NLRU	3
C	48	TCS	1	NAF	4
C	49	SRUDS	1	NLRU	3
C	50	CFJ8	1	NSUB	2
C	51	CSJ8	1	NSUB	2
C	52	CPUSE	1	NSE	5
C	53	CSESM	1	NSE	5
C	54	IH	1	NSE	5
C	55	SRUSS	1	NLRU	3
C	56	LLR	0	NAF	4
C	57	SPRTS	1	1	1
C	58	CDSE	1	1	1
C	59	NSER	0	NSE	5
C	60	CRD	1	1	1
C	61	NRC	1	1	1
C	62	RC	1	1	1
C	63	CDP	1	1	1
C	64	LCC	1	1	1
C	65	STKS	0	NLRU	3
C	66	RCY	1	1	1
C					
C	J = 14	'SET' - CALL SUBROUTINE SET			
C	J = 15	'GLOSSARY' - CALL SUBROUTINE DEFINE			
C	J = 16	'END' - TERMINATE THE PROGRAM			

```

C   J = 17 'ADJLCC' - CALL SUBROUTINE ADJUMP
C
      KOUNT(2)=NSUB
      KOUNT(3)=NLRU
      KOUNT(4)=NAF
      KOUNT(5)=NSE
C
C   OUTPUT DESIRED
C
      56 IF (NOTHER .EQ. 0) WRITE(*,58)
      58 FORMAT(/, ' REPORT?      ')
      CALL INP(4,OUTS,0,0,NMAX,NMAX,J,*1000,*56)
      IF (J .GE. 18) GOTO 59
      IF (J .EQ. 16) GOTO 999
      IF (J .EQ. 17) GOTO 140
      IF (J .GE. 14) GOTO 150
C
C   SET MASK LENGTH FOR R+M OUTPUTS.
C
      MAX=7
      IF (J .GT. 10) MAX=5
      GOTO 60
C
C   SET POINTERS AND MASK LENGTH FOR COST OUTPUTS.
C   ICODE = OUTPUT CODE (10-58)
C   IND = INDEX INTO JTYP (1-49)
C   KTYP = 1 - SCALAR OR COST
C           2 - SUBSYSTEM
C           3 - LRU
C           4 - AFSC
C           5 - SE
C
      59 ICODE=J-8
      IND=ICODE-9
      KTYP=JTYP(IND)
      NTOTAL=KOUNT(KTYP)
      IF (KTYP .EQ. 1) GOTO 100
      MAX=7
      IF (KTYP .EQ. 4 .OR. KTYP .EQ. 5) MAX=5
C
C   SET MASK.
C
      60 IF (NMASK .EQ. -1) GOTO 65
      NUM=NMASK
      GOTO 75
      65 IF (NOTHER .EQ. 0) WRITE(*,70)
      70 FORMAT(/, ' MASK=      ')
      CALL INP(5,0,XMASK,0,MAX,50,NUM,*1000,*65)
C
C   DUMP R+M OUTPUTS FROM /RAM/.
C
      75 IF (J .GE. 18) GOTO 100

```

```

      IF (J .GT. 8) GOTO 80
C
C SUBSYSTEM DATA
C
      CALL DUMP(IO1,IO2,SDAT(1,J,1),SDAT(1,J,2),NSUB,JADD(J),XMASK,
&  NUM,SEQID,7,OUTS(J))
      GOTO 56
C
C LRU DATA
C
      80 JND=J-8
      IF (J .GT. 10) GOTO 90
      CALL DUMP(IO1,IO2,UDAT(1,JND,1),UDAT(1,JND,2),NLRU,JADD(J),XMASK,
&  NUM,LEQID,7,OUTS(J))
      GOTO 56
C
C AFSC DATA
C
      90 JND=J-10
      CALL DUMP(IO1,IO2,ADAT(1,JND,1),ADAT(1,JND,2),NAF,JADD(J),XMASK,
&  NUM,AFID,5,OUTS(J))
      GOTO 56
C
C NOW COST OUTPUTS FROM IO3A AND MAYBE IO4A. FIRST REWIND AND
C BYPASS HEADER RECORD.
C
      100 K=0
      REWIND 14
      READ(14) DUMA
      READ(14) DUMN
      IF (IO4A .EQ. 0) GOTO 110
      REWIND 15
      READ(15) DUMA
      READ(15) DUMN
C
C READ FROM OUTPUT FILE(S) UNTIL CODE ON FILE MATCHES REQUESTED
C CODE.
C
      110 READ(14) I,VAL1
      IF (I .EQ. 0) GOTO 990
      IF (IO4A .NE. 0) READ(15) I,VAL2
      IF (ICODE .EQ. 1) GOTO 120
      GOTO 110
C
C TRANSFER DATA INTO STORA (AND STORB). GET MORE IF NECESSARY.
C
      120 K=K+1
      STORA(K)=VAL1
      IF (IO4A .NE. 0) STORB(K)=VAL2
      IF (K .LT. NTOTAL) GOTO 110
C
C CALL DUMP WITH CORRESPONDING PARAMETERS BASED ON KTY (SINGLE

```

```

C  VALUE, SUBSYSTEM, ETC.).
C
      GOTO (121,122,123,124,125),KTYP
      STOP 1234
121 CALL DUMP(IO3A,IO4A,STORA,STORB,1,0,XMASK,0,BLANK,6,OUTS(J))
      GOTO 56
122 CALL DUMP(IO3A,IO4A,STORA,STORB,NSUB,JTOT(IND),XMASK,NUM,SEQID,7,
      &  OUTS(J))
      GOTO 56
123 CALL DUMP(IO3A,IO4A,STORA,STORB,NLRU,JTOT(IND),XMASK,NUM,LEGID,7,
      &  OUTS(J))
      GOTO 56
124 CALL DUMP(IO3A,IO4A,STORA,STORB,NAF,JTOT(IND),XMASK,NUM,AFID,5,
      &  OUTS(J))
      GOTO 56
125 CALL DUMP(IO3A,IO4A,STORA,STORB,NSE,JTOT(IND),XMASK,NUM,SEID,5,
      &  OUTS(J))
      GOTO 56

C
C  EXTRACT ADJUSTED LCC AND DUMP.  -1 IN CALL INHIBITS SORT.
C
140 CALL ADJUMP(IO3A,IO4A,STORA,STORB,TEMP,NADJ,*145)
      CALL DUMP(IO3A,IO4A,STORA,STORB,NADJ,1,0,-1,TEMP,3,OUTS(J))
      GOTO 56
145 WRITE(*,146)
146 FORMAT(/,' INSUFFICIENT DATA TO COMPUTE ADJLCC')
      GOTO 993

C
150 IF (J.EQ. 14) GOTO 160
      CALL DEFINE
      GOTO 56
160 CALL SET
      GOTO 56

C
990 WRITE(*,992)
992 FORMAT(/,' NOT COMPUTED.')
993 NOTHER=0
      GOTO 56
999 JABT=1
1000 RETURN
      END
      SUBROUTINE DUMP(IO1,IO2,A,B,NTOT,JTOT,XMASK,NUM,ID,LID,HEAD)

C
C  THIS ROUTINE, CALLED BY OUTPUT, PRINTS ARRAYS A AND B TO THE
C  USER.
C
C
C      IO1 - BASE FILE
C      IO2 - PERTURBED FILE (OR ZERO IF NONE)
C      A - BASE DATA
C      B - PERTURBED DATA
C      NTOT - SIZE OF A (AND B)
C      JTOT - 0 - DON'T TOTAL LIST

```

```

C      - 1 - TOTAL LIST
C      MASK - MASK FOR PRINTING
C      NUM - LENGTH OF MASK
C      ID - EQUIPMENT NAMES
C      LID - LENGTH OF ID
C      HEAD - OUTPUT NAME TO BE DISPLAYED
C
C      INTEGER JFLAG(120),IND(120)
C      REAL A(1),B(1),X(120,3),TOT(2)
C      CHARACTER*1 XMASK(10),IDS(10)
C      CHARACTER*7 ID(1),XID(120)
C      CHARACTER*10 HEAD,PLIB,PC
C      CHARACTER*10 BL,CHA,DIF,DASH,TOTL,OUT1,OUT2
C      CHARACTER*13 FIELD(3)
C      INTEGER MASK,TITLE
C      COMMON/ALL/NOTHER,KPR,KSO,KLI,NMASK,NTITL,MASK(10),TITLE(10)
C
C      DATA BL /'          '/
C      DATA CHA /'% CHANGE '/
C      DATA DIF /'DIFFERENCE'/
C      DATA DASH/'-----'/
C      DATA TOTL/'TOTAL    '/
C      DATA OUT1/'BSEOUT   '/
C      DATA OUT2/'PRTOUT   '/
C
C      N=0
C      TOT(1)=0.0
C      TOT(2)=0.0
C      XMAX=1.E-11
C      IF (IO2.EQ. 0) GOTO 3
C
C      IF (KPR.EQ. 0) GOTO 1
C      JP=KPR
C      GOTO 3
C      1 IF (NOTHER.EQ. 0) WRITE(*,2)
C      2 FORMAT(/,' DO YOU WANT: ',/,
C      *          ' 1 - % CHANGE',/,
C      *          ' 2 - DIFFERENCE ?  ')
C      CALL INP(1,0,0,1,2,170,JP,*230,*1)
C
C      LOOP THROUGH DATA, MOVING A AND B TO X(N,1) AND X(N,2). SET
C      X(N,3) TO DIFFERENCE OR % CHANGE (IF PERTURBING). TOTAL AS
C      WE GO.
C
C      3 DO 6 J=1,NTOT
C      IF (NUM.LE. 0) GOTO 5
C      READ(ID(J),4) (IDS(M),M=1,LID)
C      4 FORMAT(7A1)
C      CALL MATCH(NUM,XMASK,LID,IDS,*6)
C      5 N=N+1
C      AJ=A(J)
C      XMAX WILL DEFINE FORMAT.

```

```

XMAX = AMAX1(XMAX,ABS(AJ))
TOT(1) = TOT(1)+AJ
X(N,1)=AJ
XID(N)=ID(J)
IND(N)=N
IF (IO2 .EQ. 0) GOTO 6
BJ=B(J)
XMAX = AMAX1(XMAX,ABS(BJ))
TOT(2)=TOT(2)+BJ
X(N,2)=BJ
TEMP=BJ-AJ
IF (JP .EQ. 1) TEMP=TEMP*100. / (AJ+1.E-20)
X(N,3)=TEMP
6 CONTINUE

C
C NUM < 0 IS SPECIAL CASE TO INHIBIT SORT OR IF ONLY ONE ITEM,
C DON'T SORT.
C
    IF (NUM .LT. 0) GOTO 160
    IF (N-1) 7,160,9
7 WRITE(*,8)
8 FORMAT(' NO DATA')
NOTHER=0
RETURN

C
C DETERMINE HOW TO SORT.
C
9 IF (KSD .EQ. -1) GOTO 10
IAS=KSD
GOTO 30
10 IF (NOTHER .EQ. 0) WRITE(*,20)
20 FORMAT(/, ' SORTED? ')
CALL INP(3,0,0,0,0,210,IAS,*230,*10)
30 IF (IAS .EQ. 0) GOTO 160
40 IF (NOTHER .EQ. 0) WRITE(*,50)
50 FORMAT(/, ' ASCENDING? ')
CALL INP(3,0,0,0,0,220,IAS,*230,*40)

C
    IN=1
    IF (IO2 .EQ. 0) GOTO 80
60 IF (NOTHER .NE. 0) GOTO 74
    WRITE(*,70)
70 FORMAT(/, ' SORT ON: ',/,
*          ' 1 - BASE DATA',/,
*          ' 2 - PERTURBED DATA')
    IF (JP .EQ. 1) WRITE(*,71)
71 FORMAT(' 3 - % CHANGE ? ')
    IF (JP.EQ.2) WRITE(*,72)
72 FORMAT(' 3 - DIFFERENCE ? ')
74 CALL INP(1,0,0,1,3,150,IN,*230,*60)
    JABS=0
    IF (IN .NE. 3) GOTO 80

```

```

75 IF (NOTHER .EQ. 0) WRITE(*,77)
77 FORMAT(/, ' SORT ON ABSOLUTE VALUE?      ')
   CALL INP(3,0,0,0,0,230,JABS,*230,*75)

C
C AT THIS POINT:
C   IAS = 0 - ASCENDING
C       = 1 - DESCENDING
C   IN  = 1 - SORT ON BASE
C       = 2 - SORT ON PERTURBED
C       = 3 - SORT ON DIFFERENCE OR % CHANGE
C   JABS = 0 - SORT REGULAR
C         = 1 - SORT ON ABSOLUTE VALUE
C
80 NEXT=0
   IF (IAS .EQ. 0) NEXT=N+1
   DO 90 J=1,N
     JFLAG(J)=0
90 CONTINUE

C
C SORT BY POINTER IND.
C
   DO 150 J=1,N
     XMIN=1.E30
     DO 100 K=1,N
       IF (JFLAG(K) .EQ. 1) GOTO 100
       XKIN = X(K,IN)
       IF (JABS .EQ. 1) XKIN=ABS(XKIN)
       IF (XKIN .GE. XMIN) GOTO 100
       XMIN=XKIN
       KSAVE=K
100 CONTINUE
     JFLAG(KSAVE)=1
     IF (IAS .EQ. 0) NEXT=NEXT-1
     IF (IAS .EQ. 1) NEXT=NEXT+1
     IND(NEXT)=KSAVE
150 CONTINUE

C
C SET UP HEADERS AND FLAGS
C
160 PLIB=BL
   PC=BL
   NP=1
   NT=1
   IF (I02 .EQ. 0) GOTO 190
   PLIB=OUT2
   IF (JP .EQ. 1) PC=CHA
   IF (JP .EQ. 2) PC=DIF
   NP=3
   NT=2

C
190 WRITE(*,200) HEAD,OUT1,PLIB,PC
200 FORMAT(1X,1H*,A10,1X,A10,4X,A10,4X,A10)

```

```

      LINES=0
C
C   SET FORMAT
C
      JF=ALOG10(XMAX)+3
      IF (JF.LT.1) JF=1
C
C
      DO 210 M=1,N
      CALL ABORT(LINES,*230)
C
C   GET NEXT INDEX
C
      K=IND(M)
C
C   ENCODE AND MAYBE INSERT COMMAS.
C
      DO 205 J=1,NT
      CALL CONV(T(JF,FIELD(J),X(K,J))
205  CONTINUE
C
C   FOR % CHANGE, SIMPLY ENCODE, BUT FOR DIFFERENCE, CONSIDER
C   COMMAS IN CONV.
C
      IF (NP .LT. 3) GOTO 210
      IF (JP .EQ. 2) GOTO 208
      WRITE(FIELD(3),206) X(K,3)
206  FORMAT(F12.1,1X)
      GOTO 210
208  CALL CONV(T(JF,FIELD(3),X(K,3))
C
210  WRITE(*,215) XID(K), (FIELD(J),J=1,NP)
215  FORMAT(1X,A7,3(A13,1X))
C
C   PRINT TOTAL
C
      IF (JTOT .EQ. 0 .OR. N .EQ. 1) RETURN
      DO 217 J=1,NT
      CALL CONV(T(JF,FIELD(J),TOT(J))
217  CONTINUE
C
C   AS BEFORE WITH % CHANGE OR DIFFERENCE.
C
      IF (NP .LT. 3) GOTO 219
      IF (JP .EQ. 2) GOTO 218
      XXX = (TOT(2)-TOT(1))*100.0/(TOT(1)+.1E-20)
      WRITE(FIELD(3),206) XXX
      GOTO 219
218  CALL CONV(T(JF,FIELD(3),TOT(2)-TOT(1))
C
C
219  WRITE(*,220) (DASH,J=1,NP)

```

```

220 FORMAT(11X,A10,4X,A10,4X,A10)
    WRITE(*,215) TOTL,(FIELD(J),J=1,NP)
230 RETURN
    END
    SUBROUTINE CONV (JF,FIELD,X)
C
C  CALLED BY DUMP TO ENCODE VALUE X INTO 'FIELD' ACCORDING TO
C  FORMAT JF, AND POSSIBLY INSERT COMMAS.
C
    CHARACTER*13 FIELD
    CHARACTER*1 CIN(13),COUT(13),SUNIM,COM,BL
    CHARACTER*7 VAR(8)
C
    DATA VAR/'(F13.8)', '(F13.7)', '(F13.6)', '(F13.5)',
&          '(F13.4)', '(F13.3)', '(F13.2)', '(E13.5)'/
    DATA SUNIM/'-'/
    DATA COM  /', '/
    DATA BL   /' ' /
C
    IF (JF .LE. 7) GOTO 4
    IF (JF .LE. 10) GOTO 5
    IF (JF .GT. 12) GOTO 2
C
C  CONVERT TO INTEGER AND ENCODE
C
    IX=X
    WRITE(FIELD,1) IX
    1 FORMAT(I13)
    JFROM=11
    GOTO 15
C
C  SIMPLY ENCODE AND RETURN
C
    2 JF=8
    4 WRITE(FIELD,VAR(JF)) X
    RETURN
C
    5 WRITE(FIELD,10) X
    10 FORMAT(F13.1)
    JFROM=9
C
C  DECODE INTO CHARACTERS TO INSERT COMMAS.
C
    15 READ(FIELD,20) CIN
    20 FORMAT(13A1)
C
C  COPY RIGHT THREE DIGITS (AND DECIMALS IF JFROM=9).
C
    DO 30 L=JFROM,13
        COUT(L)=CIN(L)
    30 CONTINUE

```

```

C      LIN=JFROM
      LOUT=JFROM
      K4=3
C
C      60 LEFT
C
C      40 LIN=LIN-1
      LOUT=LOUT-1
C      IF BLANK, WE'RE THROUGH.  IF MINUS, STORE MINUS AND WE'RE
C      THROUGH.
C
      IF (CIN(LIN) .EQ. BL) GOTO 60
      IF (CIN(LIN) .NE. SUNIM) GOTO 45
      COUT(LOUT)=SUNIM
      GOTO 65
C
C      BUMP DIGIT COUNTER.  IF 4 DIGITS, INSERT COMMA IN OUTPUT
C      BUFFER.
C
      45 K4=K4+1
      IF (K4.LT.4) GOTO 50
      K4=1
      COUT(LOUT)=COM
      LOUT=LOUT-1
C      COPY DIGIT
      50 COUT(LOUT)=CIN(LIN)
      GOTO 40
C
C      PAD WITH BLANKS AND ENCODE BACK INTO FIELD.
C
      60 IF (LOUT.LE.0) GOTO 70
      COUT(LOUT)=BL
      65 LOUT=LOUT-1
      GOTO 60
      70 WRITE(FIELD,20) COUT
      RETURN
      END
      SUBROUTINE ADJUMP(I03A,I04A,STORA,STORB,TITLE,LINE,*)
C
C      CALLED BY OUTPUT TO TAKE ADJUSTED COSTS OUT OF I03A (AND
C      I04A) AND FORMAT THEM FOR OUTPUT INTO STORA (AND STORB).
C
      KODE - PRINT CODE OF ADJUSTED COST:
C          80 - NON-RECURRING COSTS
C          81 - RECURRING COSTS
C          82 - DISPOSAL COSTS
C          83 - ADJLCC
C      KOUNT - SUBSCRIPT FOR ALPHA:
C          1 - NR
C          2 - RC
C          3 - DP

```

```

C   LINE - OUTPUT LINE NUMBER
C   TITLE - ARRAY CONTAINING TITLE FOR EACH LINE - COST CATEGORY
C           AND YEAR.
C
C   DIMENSION STORA(120),STORB(120)
C   INTEGER DUMA(20),DUMN(6)
C   CHARACTER*7 FIELD
C   CHARACTER*7 TITLE(30)
C   CHARACTER*2 ALPHA(3)
C
C   COMMON /SHARE/ SIN(40,4),RUIN(120,13),AFIN(50,15),SEIN(50,9),
C   *           DSE(50,2),AIDATA(50),COSTS(27),SCAL(78)
C
C   DATA ALPHA/'NR','RC','DP'/
C
C   GET BASE YEAR
C
C   IYEAR=SCAL(78)
C   IF (SCAL(78) .LE. -1.E30) GOTO 10
C   GOTO 30
10  WRITE(*,20)
20  FORMAT(/,' BASE YEAR NOT FOUND IN DATA.  SET TO 1')
C
C   IYEAR=1
30  LINE=0
C   KODE=80
C   REWIND 14
C
C   BYPASS HEADER CARDS
C
C   READ(14) DUMA
C   READ(14) DUMN
C   KOUNT=1
C   IF (I04A .NE. 0) GOTO 110
C
C   FIND ADJUSTED COSTS IF NO PERTURBED FILE.
C
C   40 READ(14) I,VAL
C   IF (I .EQ. 0) RETURN 1
C   IF (I .EQ. 83) RETURN
C   IF (I .LT. 80) GOTO 40
C   LINE=LINE+1
C   STORA(LINE)=VAL
C
C   NEXT COST CATEGORY
C
C   50 IF (I .EQ. KODE) GOTO 60
C   KODE=KODE+1
C   KOUNT=KOUNT+1
60  WRITE(FIELD,70) ALPHA(KOUNT),IYEAR
70  FORMAT(A2,1X,I4)
C   TITLE(LINE)=FIELD

```

```

        IYEAR=IYEAR+1
        GOTO 40
C
C  FIND ADJUSTED COSTS FOR REGULAR AND PERTURBED OUTPUTS.
C  BYPASS HEADER CARDS
C
110 REWIND 15
    READ(15) DUMA
    READ(15) DUMN
120 READ(14) I1,VAL1
    READ(15) I2,VAL2
C
C  END OF FILE IS REACHED.  ADJLCC NOT FOUND.  RETURN TO PRINT
C  ERROR MESSAGE.
C
125 IF (I1 .EQ. 0 .AND. I2 .EQ. 0) RETURN 1
C
C  KEEP READING CARDS UNTIL PRINT CODE FOR FIRST ADJUSTED COST
C  IS FOUND.
    IF (I1 .LT. 80) GOTO 120
C
C  IF BOTH CARDS HAVE PRINT CODE 83, WE HAVE ALL ADJUSTED COSTS.
C
    IF (I1 .EQ. 83 .AND. I2 .EQ. 83) RETURN
    LINE=LINE+1
    IF (I1 .NE. KODE) GOTO 130
    IF (I2 .NE. KODE) GOTO 150
C
C  I1 = I2 = KODE
C  BOTH FILES HAVE THE SAME COST CATEGORY.
C
126 STORA(LINE)=VAL1
    STORB(LINE)=VAL2
    WRITE(FIELD,129) ALPHA(KOUNT),IYEAR
129 FORMAT(A2,1X,I4)
    TITLE(LINE)=FIELD
    IYEAR=IYEAR+1
    GOTO 120
130 IF (I2 .EQ. KODE) GOTO 140
C
C  I1 = I2 = KODE + 1
C  BOTH FILES CHANGED AT THE SAME TIME.
C
    KODE=KODE+1
    KOUNT=KOUNT+1
    GOTO 126
C
C  I1 = KODE + 1
C  I2 = KODE
C  PERTURBED FILE HAS MORE DATA.
C
140 STORA(LINE)=0.0

```

```

        STORB(LINE)=VAL2
        WRITE(FIELD,129) ALPHA(KOUNT),IYEAR
        TITLE(LINE)=FIELD
        IYEAR=IYEAR+1
        READ(15) I2,VAL2
        GOTO 125
C
C  I1 = KODE
C  I2 = KODE + 1
C  BASE FILE HAS MORE DATA.
C
150 STORA(LINE)=VAL1
    STORB(LINE)=0.0
    WRITE(FIELD,129) ALPHA(KOUNT),IYEAR
    TITLE(LINE)=FIELD
    IYEAR=IYEAR+1
    READ(14) I1,VAL1
    GOTO 125
END

```

### Bibliography

1. Anderson D. R., D. J. Sweeney, and T. A. Williams. An Introduction to Management Science: Quantitative Approaches to Decision Making (Third Edition). New York: West Publishing Company, 1982.
2. Baer, Jean-Loup. Computer Systems Architecture. Rockville MD: Computer Science Press, 1980.
3. Bihi, A. "The Revolution is Here to Stay," CAD: Computer-Aided Design, 12: 107-114 (May 1980).
4. Blanchard, Benjamin S. Logistics Engineering and Management (Second Edition). Englewood Cliffs NJ: Prentice-Hall Inc., 1981.
5. Brennan, James R. "Design to Life Cycle Cost (DTLCC) Implementation," Journal of the Society of Logistics Engineers, 14: 27-30 (Fall 1980).
6. Briskin, Lawrence. Network Repair Level Analysis Model: User's Guide. AFALD/XRS, Wright-Patterson AFB OH, January 1984.
7. Brown, Gary D. and Donald H Selfton. "The Micro vs. the Applications Logjam," Datamation, 30: 96-104 (January 1984).
8. Cahill, Hugh E. and Richard C. Davis. "ADAM - A Computer Aid to Maintainability Design," 1984 Proceedings Annual Reliability and Maintainability Symposium. 12-16. IEEE Press, New York, 1984.
9. Chaney, Roy and Brian Johnson. "Maximizing Hard-Disk Performance," Byte, 9: 307-334 (May 1984).
10. Clark, Lt Col Thomas D., Professor, Management Sciences, Department of Operational Sciences. Personal Interview. AFIT/ENS, Wright-Patterson AFB OH, 20 April 1984.
11. "Computer-Aided Everything," Engineering-News Record, 207: 34-61 (3 December 1981).
12. Cook, Steven. "Plug in a Processor," PC World, 2: 56-59 (October 1983).
13. Delo-Stritto, Fred V., Engineer. Personal Interview. HQ AFLC/MMAQP, Wright-Patterson, AFB OH, 26 April 1984.

14. Drezner, Stephen M. and Richard J. Hillestad. Logistics Models: Evolution and Future Trends, Report No. P-6748. The RAND Corporation, Santa Monica CA, 1982.
15. Ferguson, Keith A., Operations Research Analyst. Personal Interview. AFLC/XRS, Wright-Patterson AFB OH, 16 April 1984.
16. Gage, Dr. Thomas W., Mathematical Statistician. Telephone Interview. AFLMC, Gunter AFS AL, 10 April 1984.
17. Gotwals, John K. "Processing Power on the IBM Personal Computer," 1983 ACM Conference on Personal and Small Computers. 132-142. Association of Computing Machinery, San Diego CA, 1983.
18. Graham, Lt Col James L., Chief, Analysis Division. Telephone Interview. AFSC/ALT, Andrews AFB MD, 25 April 1984.
19. Healey, Martin. "Junking the Mainframe," Datamation, 29: 120-136 (August 1983).
20. Huizenga, Charlie and Chip Barnaby. "But is It Really FORTRAN?," PC World, 2: 172-179 (February 1984).
21. Kaminker, Asher et. al. "A 32-Bit Microprocessor with Virtual Memory Support," IEEE Journal of Solid-State Circuits, SC-16: 548-557 (October 1981).
22. Kavuru, Sudha. "Modular Architecture," Byte, 8: 194-204 (June 1983).
23. Klement, Mary A. "Computer-Aided Design - A Tool For Supportability," Proceedings of the 18th Annual International Logistics Symposium. 1-6. Society of Logistics Engineers, Atlanta GA, 1983.
24. Laxon, W. R. "Selecting and Evaluating CAD Systems," CAD: Computer-Aided Design, 9: 233-237 (October 1977).
25. Leedy, Glenn. "The National Semiconductor NS16000 Microprocessor Family," Byte, 8: 53-66 (April 1983).
26. Logan, Capt Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1982 (AD-A118 042).

27. Miller, Terry, Operations Research Analyst. Telephone Interview. AFHRL/LRC, Wright-Patterson AFB OH, 11 April 1984.
28. Morris, Kenneth L., Director, Logistic Support Analysis. Personal Interview. AFLC/PTL, Wright-Patterson AFB OH, 11 April 1984.
29. Naas, E. L. and S. H. Eames. "Automation of Decision Making Logistics Support Analysis Tools," Proceedings of the 14th Annual International Symposium. 1-6. Society of Logistics Engineers, Clearwater Beach FL, 1979.
30. Nesbit, Irene S. "Move It to a Micro," Datamation, 29: 188-194 (October 1983).
31. Ogan, Capt Andrew, Logistic Supply Analyst. Telephone Interview. AF/LEYS, Pentagon, Washington DC, 9 April 1984.
32. Paulson, R. M., R. B. Waina, and L. H. Zacks. Using Logistics Models in System Design and Early Support Planning, Report No. R-550-PR. The RAND Corporation, Santa Monica CA, 1971.
33. Pechura, Michael A. "Comparing Two Microcomputer Operating Systems: CP/M and HDOS," Communications of the ACM, 26: 188-195 (March 1983).
34. Plata, E. F. "Logistics Support Analysis: Its Purpose and Scope," Logistics Direction, 4: 18-22 (January 1980).
35. ----- and C. R. Spinner. "Logistics Support Analysis Development and Implementation," Logistics Direction, 4: 23-27 (January 1980).
36. Raud, R. K. and B. G. Taum. "The State of the Art in Microcomputer Programming (A Survey)," Programmirovaniye, 5: 31-43 (September-October 1982).
37. Schwarz, Richard K. et al. Design for Maintenance in the Aerospace Industry: A Survey of Current Practices and Prospects for Integration with CAD. Human Resources Laboratory, Wright-Patterson AFB OH, December 1982.
38. Shoup, Terry E. Applied Numerical Methods for the Micro-Computer. Englewood Cliffs NJ, Prentice-Hall, Inc., 1984.

39. Singh, Ram K. "Supportability: New Direction and Future Course of LSA," Proceedings of the 18th Annual International Logistics Symposium. 1-5. Society of Logistics Engineers, Atlanta GA, 1983.
40. Sterkel, Terrance E. "Logistics Driven Design," Proceedings of the 14th Annual International Logistics Symposium. 1-4. Society of Logistics Engineers, Clearwater Beach FL, 1979.
41. Test and Evaluation of Technology for Acquiring Supportable Systems: User's Guide (Application). Contract F33615-79-C-0030 with Westinghouse Electric Corporation. Wright-Patterson AFB OH, 7 August 1982.
42. Tetmeyer, Col Donald, Chief, Human Resources Laboratory. Personal Interview. AFHRL/LR, Wright-Patterson AFB OH, 13 April 1984.
43. Wilschke, Jack. "Good News on the FORTRAN Front," Softalk, 2: 44-50 (November 1983).
44. Wood, John W., Jr. "A Logistics Risk Assessment Methodology," Proceedings of the 16th Annual International Logistics Symposium. 1-7. Society of Logistics Engineers, Seattle WA, 1981.
45. 210844. Microprocessors and Peripheral Handbook. Intel Corporation, Santa Clara CA, 1982.
46. 6936808. Technical Reference, Personal Computer XT. International Business Machines, Boca Raton FL, 1983.

## VITA

Captain Donald G. Davidson was born in New Albany, Indiana on 14 March 1951. He enlisted in the USAF in 1970. After separation from the Air Force in 1974, he enrolled in Indiana University Southeast from which he received a Bachelor of General Studies degree in January 1978. He then received a commission in the USAF through the ROTC program at the University of Louisville and entered active duty in July 1978. Assigned to 381 Strategic Missile Wing at McConnell AFB, Wichita, Kansas he served as a missile combat crew commander. Captain Davidson entered the School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB, Ohio in May 1983.

Permanent address: 617 West Spring Street  
New Albany, Indiana 47150

D-A147 666

ADAPTING LOGISTICS MODELS TO A MICROCOMPUTER FOR  
INTERFACE WITH COMPUTER... (U) AIR FORCE INST OF TECH  
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST..

3/3

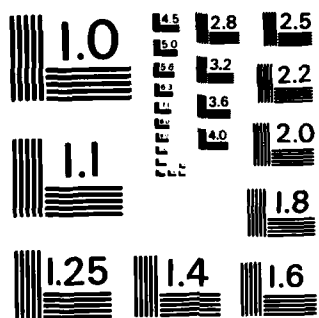
UNCLASSIFIED

D G DAVIDSON ET AL. SEP 84

F/G 15/5

NL

END



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

## VITA

Captain John J. Fraser was born on 11 July 1950 in Houston, Texas. He attended the University of Houston from which he received a Bachelor of Science in Mathematics in December 1973. Captain Fraser received his commission in the USAF through Officer's Training School in February 1975. He served eight years as a missile combat crew member at McConnell AFB, Wichita, Kansas. While stationed at McConnell AFB, he received a Master of Arts in Human Relations from Webster University, St Louis, Missouri. He entered the School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB, Ohio in May 1983.

Permanent address: 115060 Cedar Creek  
Houston, Texas 77077

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GLM/LSM/84S-11			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION School of Systems and Logistics		6b. OFFICE SYMBOL (If applicable) AFIT/LS		7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433			7b. ADDRESS (City, State and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Human Resources Laboratory		8b. OFFICE SYMBOL (If applicable) AFHRL/LRA		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433			10. SOURCE OF FUNDING NOS.		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
					WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19					
12. PERSONAL AUTHOR(S) Donald G. Davidson, B.G.S., Capt, USAF John J. Fraser, B.S., M.A., Capt, USAF					
13a. TYPE OF REPORT MS Thesis		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Yr., Mo., Day) 1984 September	
				15. PAGE COUNT 203	
16. SUPPLEMENTARY NOTATION Approved for public release; distribution unlimited. Lynn E. WOLVER Dean for Research and Professional Development Air Force Institute of Technology (AFIT) Wright-Patterson AFB, OH 45433					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary; use block number)		
FIELD	GROUP	SUB. GR.			
15	05		Computer, computer-aided design, logistics,		
09	02		maintainability, microcomputer, models		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)  Title: ADAPTING LOGISTICS MODELS TO A MICROCOMPUTER FOR INTERFACE WITH COMPUTER-AIDED DESIGN SYSTEMS  Thesis Advisor: William B. Askren, Ph.D.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL William B. Askren, Ph.D.			22b. TELEPHONE NUMBER (Include Area Code) 513-255-3871		22c. OFFICE SYMBOL AFHRL/LRA

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Logistics concerns such as reliability and maintainability are the results of product design. Logistics models are the tools used by the logistics engineers to analyze these logistics concerns. Currently, logistics models are run primarily on mainframe computers and at later stages of the design process. If logistics models were adapted to microcomputers, the models would be more accessible to the logistics engineers, thus resulting in products which are more reliable and more easily maintained.

A further step would be to interface these models with a computer-aided design (CAD) system. CAD systems have proven to be a very useful engineering tool during product design. The interfacing of these models to a CAD system would allow the logistics engineer to analyze design earlier, thus achieving greater flexibility in the design process.

This research examines the difficulties of selecting models for incorporation into a CAD system and the use of microcomputers to run these models. A selection function was developed to identify models for specific types of analysis and their suitability for incorporation into a CAD system. The literature on microcomputers was examined to determine the limitations of microcomputers to run large logistics models. To further define these limitations the Reliability Maintainability Cost Model was adapted to an IBM-PC micro-computer.

*thesis*

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

**END**

**FILMED**

**12-84**

**DTIC**